POWER AWARE DISTRIBUTED SYSTEMS

University of Southern California Marina Del Rey

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-7 has been reviewed and is approved for publication.

APPROVED: /s/

                    ANDRETTA DENNIS, 2LT., USAF
                    Project Engineer

FOR THE DIRECTOR: /s/

                    WARREN H. DEBANY, JR., Technical Advisor
                    Information Grid Division
                    Information Directorate

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE JANUARY 2004	3. REPORT TYPE AND DATES COVERED Final Aug 00 – Aug 03

4. TITLE AND SUBTITLE
POWER AWARE DISTRIBUTED SYSTEMS

5. FUNDING NUMBERS
C - F30602-00-C-0154
PE - 62301E
PR - J874
TA - 37
WU - A1

6. AUTHOR(S)
Brian Schott, Mani Srivastava,
Ronald Riley, and Igor Elgorriaga

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California Marina Del Ray
Information Science Institute
4676 Admiralty Way
Marina Del Ray California 90292-6695

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency   AFRL/IFGA
3701 North Fairfax Drive   525 Brooks Road
Arlington Virginia 22203-1714   Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-7

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Dennis L. Andretta, 2Lt., USAF/IFGA/(315) 330-2926/ Dennis.Andretta@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*
The goal of PADS was to study power aware management techniques for wireless unattended ground sensor applications to extend their operational lifetime and overall capabilities in this battery-constrained environment. The analysis included embedded systems architectures, signal processing algorithms, simulation and planning tools, and collaborative networking protocols.

14. SUBJECT TERMS
Unattended Ground Sensors, Power Aware Systems, Acoustic Beamforming, Wireless Networking

15. NUMBER OF PAGES
234

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

9 List of Addenda   65

LIST OF FIGURES

LIST OF TABLES

1   INTRODUCTION

The goal of Power Aware Distributed Systems (PADS) project was to develop and apply power aware embedded system architectures, algorithms, middleware, simulation tools, and protocols to unattended wireless ground sensor systems. The PADS project included researchers from University of Southern California (USC) Information Sciences Institute (ISI), University of California Los Angeles (UCLA), and Rockwell Science Center (RSC). The multidisciplinary team included world-class, widely-published researchers with significant industrial experience and complementary expertise in the areas of embedded systems engineering, network protocols, real-time operating systems, and DoD application experience for acoustic, seismic, and imaging sensors. The team established a number of research goals for the PADS effort:

Identify hardware knobs that can be provided by modules (radio and processor systems) that can be altered dynamically, and externally readable parameters (power, BER, signal strength, battery, etc.) that can be provided to a power-aware runtime system.

Instrument a state-of-the-art sensor node to understand power consumption in current systems. Where can we expect significant latitude in power tradeoff? Which knobs have the greatest dynamic range? What baseline will we use for comparison?

Provide operating system extensions for power management, task scheduling, and task control on individual SensIT sensor nodes.

Create reconfigurable communication modules that adapt parameters such as error control, equalization, data rate, and noise figure in real time according to channel state.

Design a distributed sensor network control middleware for power-aware task distribution and hardware/software resource utilization migration.

Incorporate power trade-off analysis tools into the SensIT platform emulator for power aware application development and scenario simulation for sensor networks.

Develop power-aware algorithms for cooperative signal processing that exploit sensor data locality, multiresolution processing, sensor fusion, and accumulated intelligence.

Integrate advanced power aware processing and communications technology into the PADS research platform as it becomes available in the PAC/C community.

The PADS project was organized into three distinct tasks. *Task 1: Architectural Approaches* examined the physical "hardware knob" aspects of the research. This included instrumentation of the baseline platform and development of a power aware communication module. The results of this task are reported in Section 2. *Task 2: Middleware, Tools, and Techniques* investigated software aspects related to real-time operating systems for power aware operation of a single node as well as collaborative networking protocols to manage energy across a sensor field. This task also developed simulation and planning tools for networked unattended ground sensors. Task 2 is covered in Section 3. Finally, *Task 3: Algorithms* studied power-awareness for signal processing algorithms used in ground sensor systems. The team looked at a representative acoustic algorithm and a representative image-processing algorithm. The algorithm studies are reported in Section 3.4. The report concludes in Section 5 with a summary of deliverables, publications, personnel, and acronyms.

2   FINAL STATUS REPORT ON
ARCHITECTURAL APPROACHES

The Architectural Approaches task of the PADS project focused on instrumentation and analysis of a baseline unattended ground sensor system to understand how battery energy is consumed. The information gathered in this study was used to develop and validate power consumption models used in the SensorSim power aware simulator. The results of this study were also used to identify power/performance "knobs" in the hardware and determine their relative impact on overall system power for intelligent algorithm development. Similarly, the identification of power/performance "knobs" in the hardware led to microsensor system architecture insights on how to expose the "knobs" to software control. The team assembled a microsensor testbed using the commercial HIDRA™ microsensor hardware developed at Rockwell Science Center. Analysis methods and results are provided in Section 2.1. The proposed power aware microsensor architecture is described in Section 2.2. Finally, a power aware software radio was prototyped and analyzed under this task. The software radio is described in Section 2.3.

2.1   Power Analysis of Rockwell HIDRA™

The PADS team assembled a research testbed using the HIDRA™ microsensor, which was developed at the Rockwell Science Center in part under the DARPA Low-Power Wireless Integrated Microsensors (LWIM) and Adaptive Wireless Arrays for Interactive Reconnaissance, Surveillance, and Target Acquisition in Small Unit Operations (AWAIRS) programs. This system has been used in numerous DoD field experiments and is representative of other microsensor platforms in the community. Shown in Figure 1, HIDRA is constructed as a modular stack of circuit boards measuring 2.5 by 2.5 inches square. It has a pair of connectors on top and bottom for inter-module communications and power supply. Initially conceived

Figure 1: HIDRA Microsensor

for mission configurability and incremental refinements of the platform, the modularity was found to be extremely useful for subsystem power analysis. The modules could be removed or isolated and instrumented separately.

The HIDRA processor module shown in Figure 2 (left) has an Intel StrongARM 1100 embedded 32-bit CPU, 1MB of SRAM, and 4MB of flash memory. The processor has a 1.5V

**Figure 2: HIDRA SA110 Processor Module (left)
and 900MHz Radio Module (right)**

**Figure 3: HIDRA Sensor Module (left)
and Power Module (right)**

2

core voltage and 3.3V I/O. The clock rate is scalable from 59MHz to 133MHz. The SA110 does not have a floating-point execution unit. Floating-point arithmetic is emulated in software. The development environment provided by Rockwell Science Center originally used the µCOS real-time operating system (RTOS). Later releases of the support software used eCOS, an open-source RTOS maintained at RedHat, Inc. The software release supplied a Hardware Abstraction Layer (HAL), a set of libraries with standard interfaces to the hardware resources of the system including the radio and sensor boards. The Rockwell-proprietary radio board shown in Figure 2 (right) transmits and receives in the 900 MHz ISM band at up to a 100kb/s data rate with a maximum range of approximately 25 meters outdoors. The transmit output power is adjustable. A number of media access control (MAC) mechanisms were supported by the HIDRA platform, including Time Division Multiplex Access (TDMA) and Collision Sense/Detect Multiple Access (CSMA/CDMA) protocol. In Figure 3 (left), the analog sensor board has five channels with 12-bit resolution. Three are high speed (60KSPS) and two are low speed (<2KSPS). The three channels are multiplexed with variable gains of 1x, 2x, 5.02x, 10.09x, and 20.12x. The other two have individual inputs with variable gains of 10x, 43.32x, 30x, 36.68x, and 49.98x. The selectable gains are tuned to specific sensors that RSC uses for this platform, such as microphones, geophones, and magnetometers. Finally, the power board Figure 3 (right) generates regulated 1.5V and 3.3V supplies from two 9V batteries or a DC adapter. The power board also provides separate analog voltage lines for the radio and sensor modules.

2.1.1 HIDRA Power Instrumentation

Run-time monitoring of the HiDRA node-level power consumption was made possible by the development of a power instrumentation board shown in Figure 4. This board was intended to be small and inexpensive enough to power-instrument a large number of sensor nodes in the field. The power measurement for the entire node was accomplished by using a low ohm value current sensing resistor in conjunction with a Burr-Brown INA138 High-Side measurement current shunt monitor. The input from this device is connected to one of four analog inputs of a PIC16C715 8-bit

Figure 4: Power Instrumentation Board

microcontroller, running at 20 MHz. Other GPIOs of the microcontroller were tied to GPIOs of the SA1100 processor. These control pins were used start and stop sampling windows from the application software. The externally powered instrumentation board consumed 15uA typical at 5V. For data logging purposes, the microcontroller interfaced with the serial port of a laptop or PDA. This instrumentation board was first used in a field text experiment at the SITEX01 experiment funded by the DARPA SensIT program.

A module-level instrumentation concept introduced in our proposal was to design a power isolation board intended to transparently plug between boards in the stack. In addition to monitoring power, the instrumentation board would have instrumented bus transactions on the connectors between modules to correlate power consumption within very small time windows.
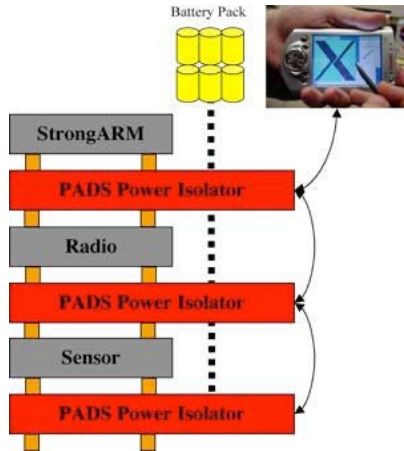
Figure 5: Module Isolation Concept

For example, a packet sent from the processor to the radio could trigger power logging. Figure 5 shows the isolation board concept. However, this approach turned out to be impractical. Unfortunately, the HiDRA connector stack was not symmetric from module to module, so a custom isolation board would have to have been developed for each module under test. The development manpower was determined to outweigh the benefits. Instead, the HiDRA subsystem analysis was performed by physically removing modules. First, the baseline application was executed on the processor entirely alone with interactions with the sensor and radio modules bypassed in the code. The serial channels to these modules were driven to include data transfer time in the analysis, but errors when communicating to the missing modules were ignored. Second, the processor and sensor modules were measured together with the radio transactions commented out. Third, the sensor module was removed and the radio was added with sensor module transactions commented out. Finally, the measurement process was repeated for the entire stack including processor, radio, and sensor. The analysis results are provided in the following section.

2.1.2 HiDRA Instrumentation Results

The power measurements of the HiDRA platform in various operational modes are given in Table 1. Further detailed measurements of power consumption on the hardware are provided with the Line of Bearing algorithms discussion in Section 3.4. For this experiment, the processor was operating at 59 MHz, which is the minimum operating frequency. The application continuously transmitted and received packets over the radio. Although the SA1100 processor has a sleep mode, this mode wasn't available in the hardware abstraction layer libraries provided by Rockwell. Even if available, the processor sleep mode isn't relevant because neither the radio nor the sensor will operate without the processor active in this architecture. As shown, the average processor power in this experiment was 360mW. In this test, the sensor module had a single seismic sensor. Compared to the other modules, the sensor module is inexpensive in terms of power at 23.3mW. The radio has the widest dynamic power range. In receive mode, the radio module consumes 391.6mW. The transmit mode varies from 410.5mW at the lowest amplification to 720.5mW at the highest.

Processor	Seismic Sensor	Radio	Power (mW)
Active	On	Rx	751.6
Active	On	Idle	727.5
Active	On	Sleep	416.3
Active	On	Removed	383.3
Active	Removed	Removed	360.0
Active	On	Tx (36.3 mW)	1080.5
		Tx (27.5 mW)	1033.3
		Tx (19.1 mW)	986.0
		Tx (13.8 mW)	942.6
		Tx (10.0 mW)	910.9
		Tx (3.47 mW)	815.5
		Tx (2.51 mW)	807.5
		Tx (1.78 mW)	799.5
		Tx (1.32 mW)	791.5
		Tx (0.955 mW)	787.5
		Tx (0.437 mW)	775.5
		Tx (0.302 mW)	773.9
		Tx (0.229 mW)	772.7
		Tx (0.158 mW)	771.5

Table 1: HiDRA Power Breakdown

This hardware power utilization analysis yielded in a number of interesting observations from a systems architecture point of view:

The transmit-to-receive power ratio ranges from 1:1 to 2:1. So, receive tends to dominate total communications power if not managed intelligently. There is a natural power/performance tradeoff between latency, hop count, and duty cycle in time-division-multiplex-access TDMA communications systems (a traditional way to duty-cycle receivers). A very low power wake-up receiver could have a significant power/performance impact even if the wake-up transmitter was highly inefficient.

Approximately 50% of the receive power in the HiDRA platform is consumed by the processor. The processor is mostly idling in this operation mode and is wasting power. A communications subsystem engineered for power awareness should contain sufficient computational resources to forward packets directly without host processor intervention. The host processor can wake up periodically to update routing tables (see Figure 6).
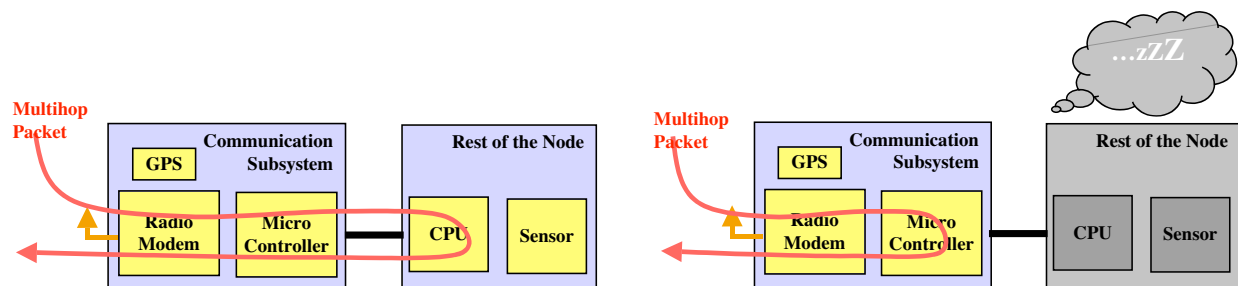
Figure 6: Power Aware Communication Subsystem

Similarly, the processor consumes approximately 90% of the power when sampling the geophone. It is mostly idling in this mode and is wasting power. A sensor subsystem engineered for power awareness should have sufficient computation resources to perform the real-time aspects of data acquisition and storage, data filtering, and possibly basic signal detection or threshold functions to trigger the main CPU. The main processor can sleep and either wakeup on a schedule or by a positive threshold event to processes a data buffer.

These systems analyses and observations clearly indicated that improved power management was possible in a microsensor platform. Even with relatively simple changes to we could save up to 50% or 90% in certain operational modes. The power aware microsensor architecture concept is described in Section 2.2.

2.2   Power Aware Microsensor Architecture

The power aware microsensor concept developed under this effort centered on the idea that a distributed "system-of-systems" architecture was a better choice for efficient power utilization. The traditional CPU-centric methodology for designing embedded systems, although it used fewer components, introduced unforeseen power consumption overhead because of the mismatch of high computational resources (a single embedded CPU) and low computational load in the most common operational modes (receiving and sensor monitoring). The modular hardware approach exhibited physically in the HYDRA platform was good, but it did not extend to power management. In order to significantly save power in a microsensor stack, the non-essential modules (for a specific operational mode) need to be in very deep sleep, or preferably, electrically disconnected from the power supply to avoid power leakage. In addition, the

HYDRA platform had no way for the software to detect what modules were present in the stack or what their operational modes were. Furthermore, the data connections in the HYDRA stack were statically defined. In order for the radio and the sensor to interact, the processor needed to be awake to broker the communication, even though it would be perfectly possible to have two microcontrollers communicate serially. A better approach would be to allow the modules to communicate independent of a central processor. Finally, some stack configurations have been contemplated that don't require a full-sized CPU for any mode of operation. It should be possible to build a node without a main processor. This modular power aware microsensor concept is detailed in Figure 7.
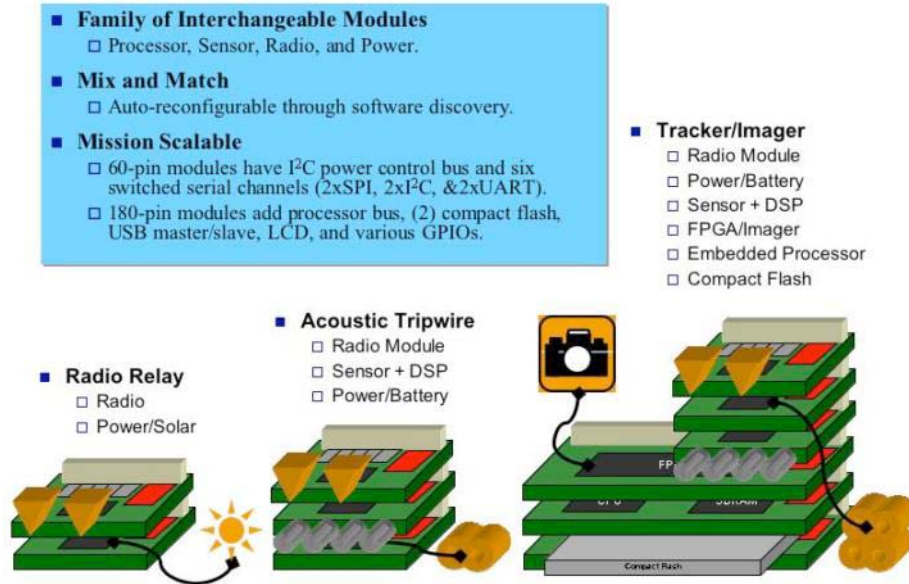
Figure 7: Modular Power Aware Microsensor

2.2.1 Implementation Goals

Implementation of a modular power aware microsensor platform as shown in Figure 7 involves compromises that balance design complexity, system flexibility, connectors and device counts, power consumption, and performance. The engineering research team debated these system-design trade-offs and arrived at a reference platform implementation that best met the requirements and system constraints. A summary of the main issues is provided:

The primary CPU in the stack should run Linux. Since most of the hard real-time signal processing would be relegated to DSPs and microcontrollers on the sensor and radio modules, the processor could afford to migrate to a robust operating system. Memory density and processor performance has advanced sufficiently to make this feasible in terms of size and power. The ISI team was involved in the development of embedded Linux for iPAQ PDAs and realizes the benefits of this immense library of software for embedded devices.

The main CPU in the stack should support commodity high-speed peripheral interfaces. Given the PDA and camera markets as technology drivers, Compact Flash (CF) and USB interfaces were placed high on the priority list. This would enable rapid prototyping with off-the-shelf hardware (cameras, microdrives, 802.11 radios, etc.) and off-the-internet device drivers. One of the frustrations of working with the HYDRA platform was the single serial

port to the outside world and dependence of building custom hardware and writing custom software for any new device.

The main CPU (or the power-hungry part of any module for that matter) should be able to be switched off at the connector to minimize power leakage when not being used. Modules should support a range of power modes from fully off to fully on. Ideally, a module should manage its own power state in response to its load and schedule.

Each module should support data interfaces appropriate to their bandwidth and power requirements. Standardizing on a common network fabric standard for all modules was considered, but this would have introduced considerable power overhead for the network interface device. Most microcontrollers, processors, and DSPs have dedicated hardware for serial interfaces (UART, SPI, and I2C) and some parallel interfaces (memory bus and cardbus). Since most of these interfaces are point-to-point, the serial connections between modules should be capable of being dynamically switched.

Modules should support software discovery and collaborate to manage power and data connections. This led to the decision to incorporate a small microcontroller on each module to act as a power controller. The power controllers communicate with each other over multi-master 2-wire I2C. They provide a low-power control network for the modules over I2C, manage the power switch to the module, and manage the switch fabric for the serial channels. A conceptual diagram of the "system of systems" approach is shown in Figure 8.
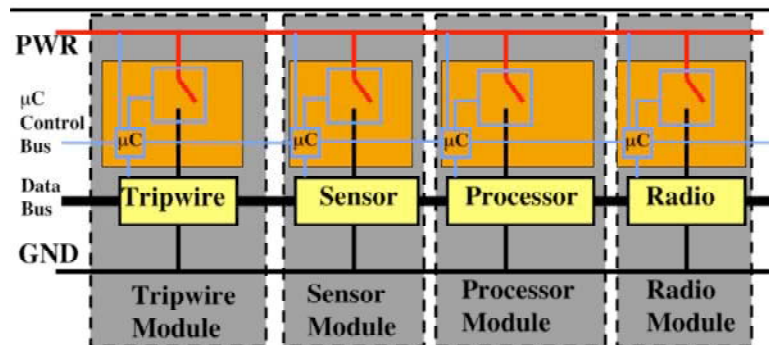
Figure 8: Distributed "System of Systems" Architecture

Initial design of the physical implementation of this power aware microsensor architecture started under PADS, but then transitioned to the DARPA PAC/C Phase II program under the ISI-led Power Aware Sensing Tracking and Analysis (PASTA) effort. The design goal was to build a compact stack of interchangeable boards (processor modules, radio modules, and sensor modules) that could expose the maximum number of hardware power "knobs" to software control. A family of modules would be developed to explore different aspects of power management in embedded systems. The physical dimensions selected were deliberately aggressive. The board footprint was just large enough for the stack connector, and a Compact Flash (CF) socket. The other boards developed would adhere to this footprint. A 180-pin stack connector was selected to accommodate a parallel memory bus and several serial channels. A 60-pin connector in the same family was plug-compatible with the 180, so the connector standard was subdivided into a 60-pin region for serial channels and a 120-pin region for the parallel busses. The 60-pin region 48 pins allocated to six 8-bit "serial" channels. Two each of the channels were designated as I2C, UART, and SPI. However, the 8-pins per channel could be used for any purpose. The core 60-pin region also contains the I2C control network for the

power microcontrollers, power pins, and some miscellaneous control pins. Where the 60-pin connector is rigidly standardized for any module, the processor module defines the allocation of the 120-pin "parallel bus". As shown in Figure 9, in the UP direction from the processor card, the pins are allocated to the processor memory bus, LCD panel, USB Master and Slave, MMC/SD card interface, and audio CODEC. In the DOWN direction relative to the processor, the pins are allocated as two Compact Flash ports and some debug pins.
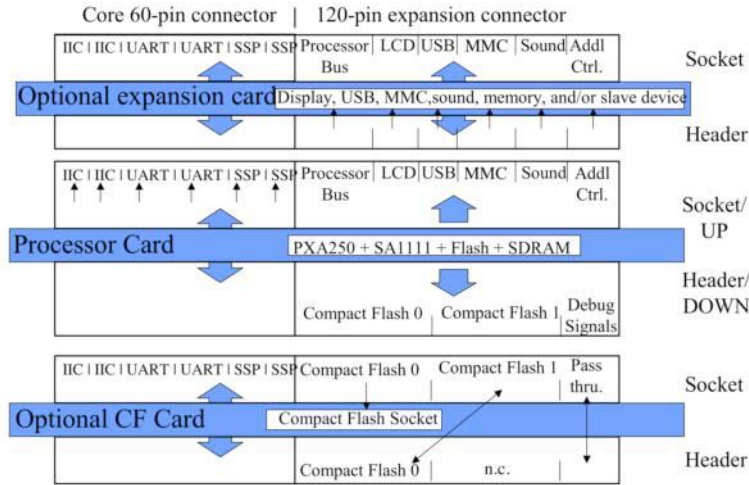
Figure 9: Module Stack Pin Allocation

Figure 10 shows the power microcontroller and how the processor module is wired to the serial channels. The module power microcontroller has an I2C (IIC) control network interface along with clock and reset pins. The regulated 3V power pins supply power to the power microcontroller, but the remainder of the module resides behind a power switch. The six serial channels are wired to native interfaces of the PXA250 processor. Bus isolation switches have been added so that the microcontroller can disconnect the module when a channel isn't being used, such as when the module is powered down.
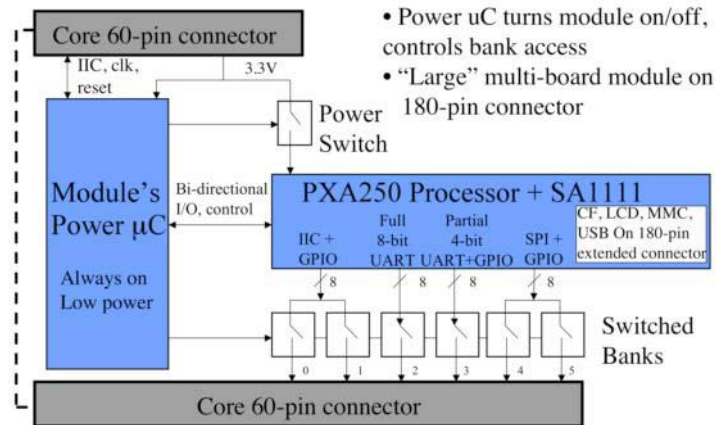
Figure 10: Example PXA25x Module

Other modules in the stack are wired in a similar fashion. A channel can be "allocated" by the power microcontrollers by turning on switches on the two communicating modules. The two module devices can then interact directly using their own protocol (SPI, UART, etc.). This arrangement introduces a minimum number of extra components yet enables dynamic channel

8

allocation. It also allows individual modules to be completely disconnected from the power supply (except for the very low overhead of a power microcontroller and the isolation switches).

2.2.2 Implementation Results

The processor selected for this effort is the Intel PXA25x™ XScale family of embedded processors commonly used in cell phones and personal digital assistants. It is a 32-bit StrongARM™ instruction set architecture device that operates between 100MHz and 400MHz. As with the StrongARM, floating point support is emulated in software. The processor supports dynamic core voltage scaling from 0.95V to 1.5V. The processor module includes 32MB of Flash and 64MB of Mobile SDRAM. The SDRAM has a programmable number of refresh banks to save power while in sleep mode. The module also includes the Intel SA1111 peripheral co-processor for USB Master support, two Compact Flash ports, and other peripheral interfaces. Figure 11 shows the power breakdown for the PXA250 processor module. The outer graph shows a dynamic operational power range from 100MHz idling at 200mW to full utilization at 400MHz with 100% memory accesses at about 1.5W. The middle graph shows power dissipation between 2mW and 7mW depending on the amount of SDRAM that is refreshed. The inner graph shows about a 0.1mW power overhead for the power microcontroller and associated switches. This analysis demonstrates that power "knobs" with a wide dynamic range can be added to a microsensor system without dramatically changing the basic architecture
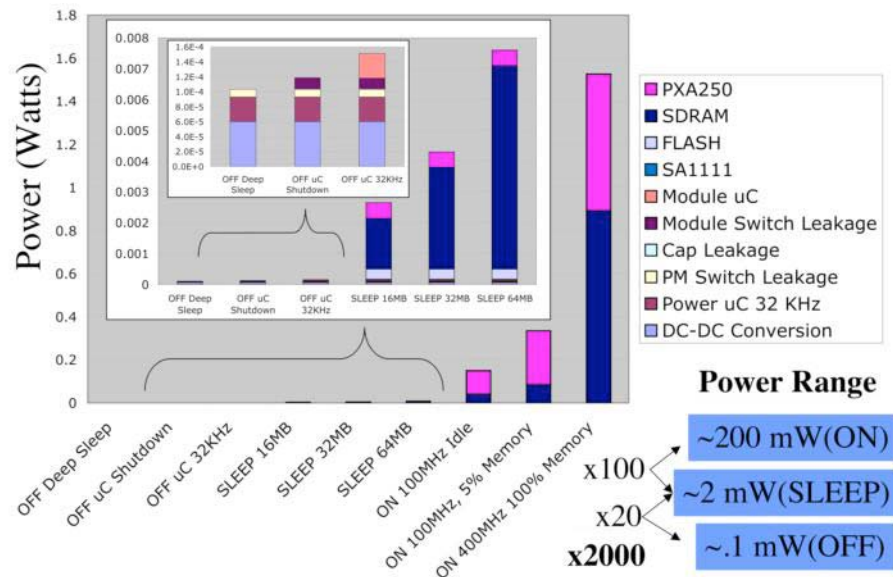
Figure 11: PXA25x Power Breakdown by Mode

2.2.3 Phase 2 Status Update

The follow-on Phase 2 PASTA effort has undertaken construction of this microsensor and is validating the results in laboratory and field experiments. A brief summary of the node development status from the PASTA effort is given here because it shows a reduction to practice of the concepts developed under the PADS project. Figure 12 shows the PASTA power aware microsensor. By the end of the PADS project (August 2003), ISI had developed the PXA255

9

processor module, a four-channel analog-to-digital-converter (ADC) module, a Compact Flash adapter module, and a power/interface board (not shown).

Total Size: 2.5" x 1.75" x ~1.5"

- **Intel PXA255 Processor Module**
  - 100-400MHz, 0.95V-1.5V core
  - Support for (2) Compact Flash modules.
  - 64MB Mobile SDRAM, 32MB FLASH.
  - Onboard power controller.
- **Compact Flash Adapter Module**
- **Four Channel ADC Module**
  - Up to 200ksps streaming, 100ksps buffered.
  - Programmable gain control and variable-cutoff AA filter.
  - Cygnal C8051F125 8-bit microcontroller @ 0-100MHz.

Figure 12: PASTA Power Aware Microsensor

Figure 13 demonstrates how the switched serial channels (SPI in this case) are used to route data from the ADC module to the PXA255 module. In the PASTA June 2003 experiment, the ADC module operated continuously while the processor slept about 97% of the time.
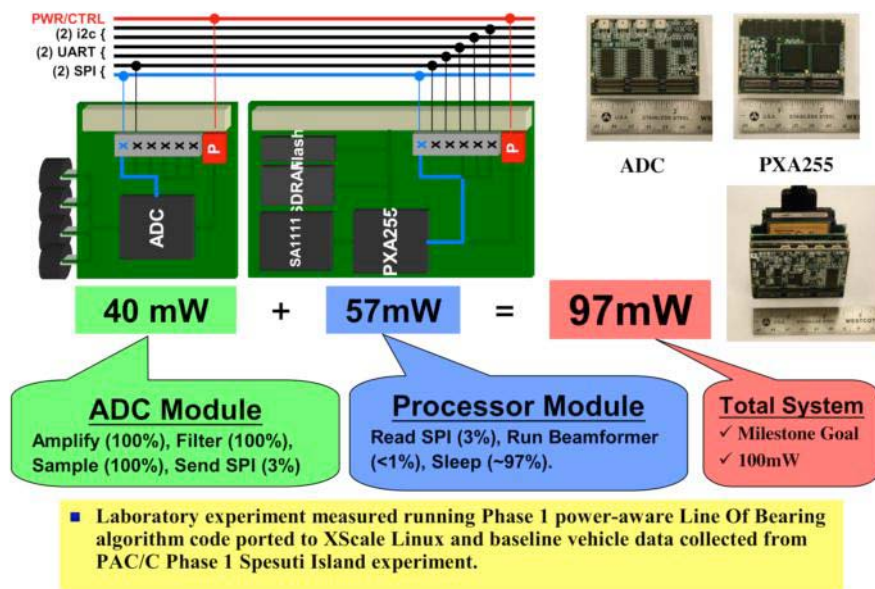
Figure 13: PASTA Experimentation Results, June 2003

2.3   RSC Power Aware FPGA Radio

The purpose of this report is to explain the development of a reconfigurable, flexible, spread spectrum modem as part of a power-aware computing and communication system (PAC/C) requested by ISI for the PAC/C community. The major requirement for such a device is to be able to dynamically adapt communication processing during runtime according to varying external conditions. Provide reconfiguration controls to middleware for power aware protocol, allowing reconfiguration not only at the lower physical layers, but also at the higher layers, such as link and network. The task presented to Rockwell Scientific to design an energy efficient, dynamically reconfigurable spread spectrum modem is quite complex without a hardware platform being defined. In order to be able to complete this task Rockwell Scientific proposed a parallel path of action. The development of the required modem optimized for an intermediate testbed platform in parallel to the work in progress of the PAC/C community to define the hardware platform. A modem with the dynamic reconfiguration capabilities with well-specified knobs, and architecture optimization for energy efficiency has been developed, tested and debugged in this testbed platform. The designed modem specifications are shown in Table 2.

Table 2: PAC/C Spread Spectrum Modem Specifications

Type of data modulation	DBPSK, DQPSK
Spreading	User selectable spreading with 0dB, 11.7dB, 14.9dB, 18.0dB and 21dB processing gain.
Data rates	User selectable from 2.1Mbps for 0dB processing to 8.4Kbps for 21dB processing gain.
PN sequences	Maximum Selectable Length of 15, 31, 63, 127 phases.
Overhead length (preamble + configuration)	78 bits with DBPSK encoding, scrambling and 11.7dB processing gain. 1103.7µs.
Payload length	Selectable from 1 Byte to 256 Bytes.
Scrambler	Selectable ON or OFF.
Interleaver	Selectable ON or OFF.
Input clock frequency	25.6 MHz.
Chipping frequency	1.067 MHz.
Number of samples per chip (Receiver mode)	4 samples per chip
Sample frequency (RX mode)	4.27 MHz.

According to the requirements, the modem should provide different configuration modes, which can be set by externally selecting from a set of clearly defined knobs. The modem should be re-configurable for variable data rates, processing gains, selectable modulation and demodulation schemes. The set of knobs available to the user are displayed in Table 3. Where data rates are selected by a combination of different knobs.

Table 3: Available Radio Knobs

Processing Gain	0dB, 11dB, 15dB, 18dB and 21dB
Scrambler	ON/OFF
Interleaver	ON/OFF
Modulation/Demodulation	DBPSK/DQPSK
Payload Length	From 1 Byte to 256Bytes (2.048Kb)

Data Rate	Selected by processing gain and modulation scheme.

The test platform implemented by Rockwell Scientific consists of a board with all the elements necessary to perform testing and debugging without a form factor in mind. This platform allowed us to make architectural optimizations, leaving hardware dependent optimization for later development once the final target platforms has been defined. At that, point hardware dependent parameters can be optimized without major architectural changes. The characteristics of the test platform are depicted in Table 4.

Table 4: Testbed Platform Characteristics

Type of FPGA	Virtex-E from Xilinx (XCV1600E)
Number of gates	1,600,000
Clock frequency	25.6 MHz
Processor	SA-1110
Processor – FPGA interface	Memory mapped
SRAM	2 MB
FLASH	16 MB
EEPROM	3 x XC18V04 (Xilinx). 12Mb

2.3.1 Modem configuration

The major requirement for this project is to have a highly dynamically reconfigurable modem. The configuration for each transmission can be changed according to the current channel conditions or user needs. The modem configuration is performed by two different methods. The transmitter is directly configured by the controlling unit, in the test bed case the SA-1110. The receiver is dynamically configured with the data being received through the channel. The receiver extracts the reconfiguration parameters and once the data has been found error free, the receiver modem dynamically re-configures itself for reception with those configuration parameters.

2.3.1.1 Configuration parameters

The configuration parameters allow the user to adapt the transmission to get the maximum throughput, maximum reliability for the conditions of the channel, or maximum energy savings for the transmission. Two encoding/decoding methods can be selected Differential Binary Phase Shift Keying (DBPSK) or Differential Quadrature Phase Shift Keying (DQPSK). Section 2.3.3 explains the choice of these two encoding schemes rather than using higher density PSK signals. Processing gain can be adjusted depending on the channel conditions and desired energy consumption. Higher processing gain allows transmission across hostile channels trading off data rate as well as energy consumption. Combining different methods of encoding and decoding as well as processing gains we indirectly select among different data rates. For added security a scrambler/de-scrambler can be turned on or off. An interleaver has been implemented which can be turned on or off to allow the spreading of burst errors to allow error correction schemes to be more efficient.

2.3.1.2 Packet format

The Phy layer packet format is defined to work with the MAC management entity. Each packet transmission has to go through code acquisition, and bit synchronization. This structure makes

the modem ideal for burst type networks. Each packet contains a preamble, a configuration header and a payload of data. The preamble as well as the header is spread with a 15-length maximum length (ML) PN sequence, scrambled and DBPSK encoded.

Preamble	Header	Payload
30 bits	48 bits	Variable number of bits (programmable)
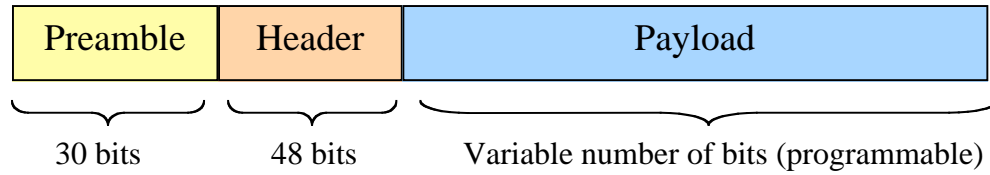
Figure 14: Packet Format

2.3.1.2.1 Preamble

The preamble is divided in two parts, synchronization (SYNC) and delimiter fields. It serves two purposes, code acquisition (SYNC) and bit synchronization (delimiter). The SYNC portion of the packet consists of 14 scrambled "0" bits. This field is provided so the receiver can perform the necessary operations for code acquisition. The receiver searches for the phase of the received PN sequence, and once acquisition is successfully performed, both received and the locally generated codes are phase aligned allowing data demodulation. The delimiter field is used to indicate the receiver modem start of the Header, the bit synchronization function. The delimiter consists of a 16-bit field [1111 0011 1010 0000], where the LSB is transmitted right after the end of the SYNC field.
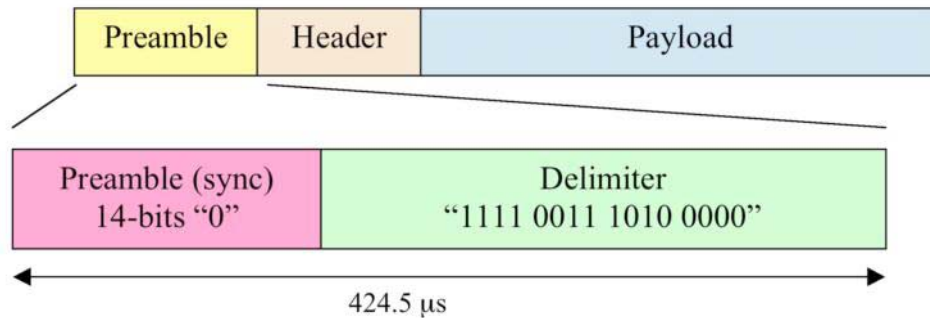
Figure 15: Preamble Format

2.3.1.2.2 Header

The header contains configuration parameters for the transmission of the payload. This field is extracted by the receiver, which then reconfigures the demodulator. The header is composed of 32 scrambled; CRC protected, DBPSK encoded bits, and spread with a ML PN sequence of length 15 chips/bit. The header is divided into four fields; signal, service, length and a CCITT CRC-16 frame check sequence.

13
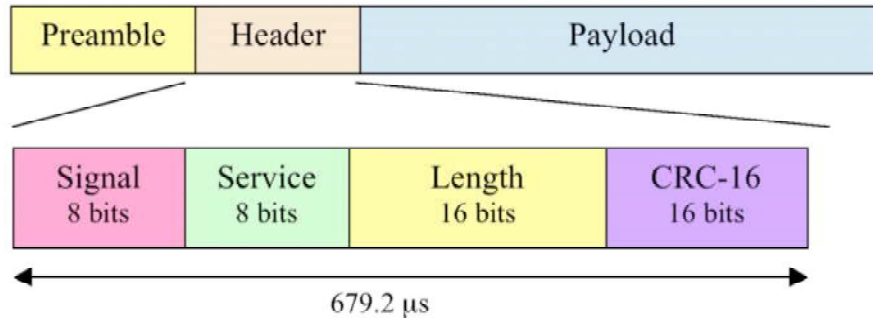
Figure 16: Header Packet Format

2.3.1.2.3 Signal field

The signal field consists of 8 bits, which contain the PN code to be used for the spreading of the payload. The CRC-16 protection starts with the first bit of the signal field.

1.   X'83 (msb to lsb) for no spreading 1-chip/bit (0dB processing gain).
2.   X'98 (msb to lsb) for 15-chips/bit spreading (11.7dB processing gain).
3.   X'94 (msb to lsb) for 31-chips/bit spreading (14.9dB processing gain).
4.   X'86 (msb to lsb) for 63-chips/bit spreading (18.0dB processing gain).
5.   X'83 (msb to lsb) for 127-chips/bit spreading (21.0dB processing gain).

2.3.1.2.4 Service Field

Service field consists of 8 bits, containing information about the spreading length, type of encoding used, scrambler, and interleaver states. From the information in this field we can extract the data rate used for the payload as shown in Table 6.

Table 5: SERVICE Field Definitions

b7	b6	b5	b4	b3	b2	b1	b0
No spreading 1=enable 0=disable	15 cpb spreading 1=enable 0=enable	31 cpb spreading 1=enable 0=disable	63 cpb spreading 1=enable 0=disable	127 cpb spreading 1=enable 0=disable	Interleaver 0 = OFF 1 = ON	Scrambler 0 = OFF 1 = ON	Encoding 0=DBPSK 1=DQPSK

When no spreading is selected a ML PN-sequence of length 127 is generated and mixed with the data at a 1-chip/bit rate. This serves as an extra security measure by scrambling the data with the PN sequence.

2.3.1.2.5 Length field

The length field is composed of 16 bits. This field is used to indicate to the receiver modem the size of the payload. This field is divided into two Bytes. The second Byte (bits 8 to 15) is reserved for future expansion. The first Byte contains the number of Bytes used for the payload. This number can be programmed from 0 to 255. The maximum payload size is 256 Bytes (2.048kbits), and the minimum payload size is 1 Byte (8 bits).

2.3.1.2.6 CCITT CRC-16 field

The CRC-16 field protects the signal, service and length fields from errors occurring during the transmission. The protection is implemented by the polynomial

$$x^{16} + x^{12} + x^{5} + 1$$

The protection starts with the first transmitted bit of the Signal field and ends with the last transmitted bit of the length field. Once the length field has been transmitted, the computed CRC code is appended to the packet.

2.3.1.2.7 Payload

The payload contains the encoded, spread, interleaved and scrambled data sent by the processor according to the header field. The length of the field depends on all the configuration parameters described in Section 2.3.1.2.2. Using the different combinations the following data rates can be achieved.

Table 6: Available Payload Data Rates

Spreading	1 chip/bit	15 chips/bit	31 chips/bit	63 chips/bit	127 chips/bit
DBPSK	1.06 Mbps	71.1 Kbps	34.4 Kbps	16.9 Kbps	8.4 Kbps
DQPSK	2.13 Mbps	142.2 Kbps	68.8 Kbps	33.8 Kbps	16.8 Kbps

2.3.2 PAC/C modem architecture

This chapter provides specific information regarding the modem architecture. First a description of the overall modem characteristics will be presented. Later a detailed description of major individual blocks is provided. The modem is divided into major functionality blocks; transmitter, acquisition and demodulation. The most complex block is the receiver, thus the report is focused in the receiver architecture.

2.3.2.1 Modem overview

The purpose of the modem is to process data in order to successfully transmit it via a wireless channel. In order to provide the most flexibility the configuration parameters are embedded in the transmission packet. The use of embedded configuration data allows change of parameters without previous exchange of these parameters, thus reducing the network overhead, and energy consumption. This scheme assumes the transmitter entity has knowledge of the conditions of the channel, thus it makes the appropriate decision on what parameters to use for maximum efficiency in terms of throughput, lower energy consumption, etc. The entire modem is implemented as a state machine. The three major states are transmit, acquisition, and demodulation. Within these blocks sub-state machines are implemented. For instance in the acquisition block, code acquisition, bit synchronization and code extraction sub-states are implemented. The division of the entire modem into different states allows implementation that maximizes the energy consumption efficiency. For example the state is to receive and sub-state to bit synchronize. The transmitter, code acquisition, configuration extraction and data demodulation blocks are disabled, thus not consuming any energy due to switching signals.

2.3.2.2 Transmitter part of the modem

The functional diagram of the transmitter portion of the modem is shown in Figure 17. At the beginning of the time slot the preamble is generated according to the configuration parameters programmed by the processor. This preamble consists of the sync, delimiter and header, as shown in section 2.3.1.2. The sync sequence is generated by scrambling and DBPSK encoding "0" bits. The length of this Sync sequence is 14 bits, which are spread at a rate of 15 chips/bit. This sequence of bits is intended to provide the receiver modem with the information

to process and perform code acquisition. Immediately following the Sync sequence the transmitter sends the delimiter [1111 0011 1010 0000] LSB to MSB. The delimiter provides bit synchronization, which signals the boundary between the preamble and the configuration data (header).
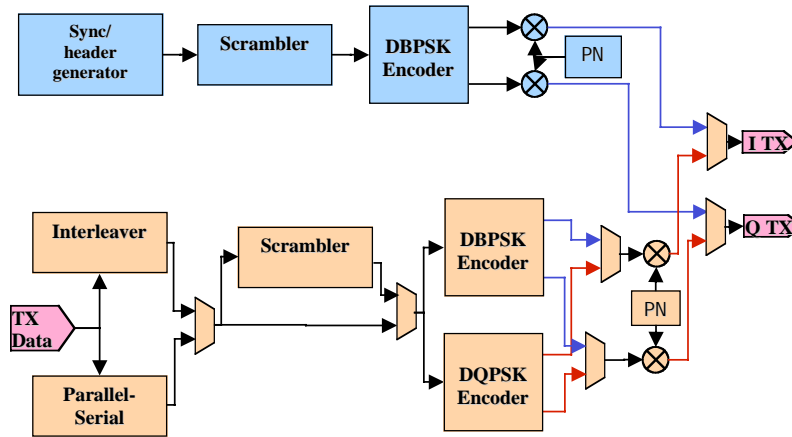
Figure 17: Functional Diagram of Transmitter Portion of the Modem

The values for the signal and service fields are read from the modem-processor interface, and directly encapsulated into the header field. The CRC-16 is dynamically computed as the bits for the header are transmitted. Once the last bit of the service field has been transmitted, the CRC-16 code is appended to the transmission as part of the header.

Figure 17 shows two independent paths for transmission, on top the preamble and header data paths, and on the bottom the payload data path. Each of these paths uses independent PN sequence generators. The top generator is used for the spreading of the preamble and header. It uses a maximal length (ML) PN sequence of length 15, with polynomial coefficients [001 1000] and initial state [111 1111]. The payload data PN generator is used for the payload spreading, the code for the polynomial used is transferred in the signal field, and the initial state is [111 1111].

The payload data transmission path in Figure 17 shows clearly the different configuration paths according to the configuration parameters selected. Before the transmitter is enabled, the processor loads the entire payload data into the Interleaver unit or parallel to serial unit. These units consist of a series of RAM blocks. Once the data has been loaded, the modem is placed into transmission mode. When the interleaver is enabled, and once the preamble has been sent, the modem will read the data by columns as shown in section 2.3.2.2.1. When the interleaver is disabled, the data will be read directly from the RAM blocks in parallel format and converted to serial format as shown in section 2.3.2.2.2.

The scrambler processes the data in a serial one-bit manner. The data scrambler initial state is [110 1100], and its architecture is shown in section 2.3.2.2.3. The outcome of this process is a serial bit output, which is encoded by either of the PSK encoders following the procedure described in section 2.3.3. After the encoder the data has already been processed and it is ready for spreading with the PN sequence. The PN sequence generator structure is shown in section 2.3.2.2.4, which for the payload data uses a variable length. The mixing (spreading) function is implemented by XOR-ing (modulo two addition) of the encoded data, and the PN sequence.

The output of the transmitter is selected between the two described paths. When the transmitter is in the preamble and header transmission state, the top path is selected for ITX and QTX. Following the transmission of the header the payload data path is selected.

2.3.2.2.1 Interleaver unit

The interleaver unit is a data communication technique used to distribute burst type errors. The interleaver re-arranges the order in which the data is transmitted. Interleaving techniques randomize and spread the location of burst errors. This spread of burst errors, combined with error correction techniques provides better results for data correction.
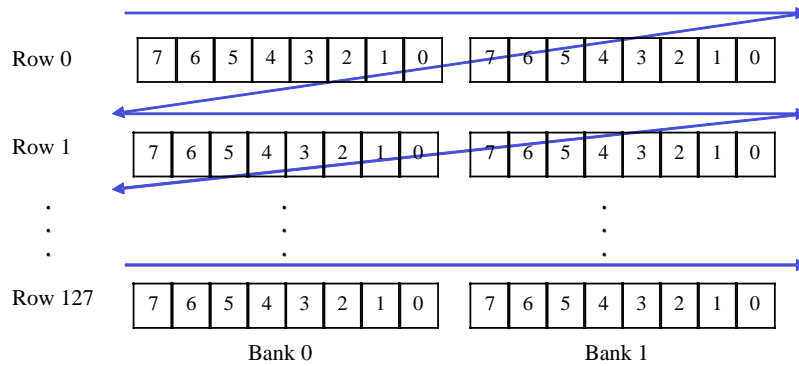
Figure 18: Interleaver row wise Byte write operation.

The interleaver unit reads data in parallel (8-bit wide) format directly from the processor. The data is preloaded into the interleaver RAM one Byte at the time (Figure 18). Once all the data has been preloaded, the modem is ready to be placed in transmission mode. Before the modem is done transmitting the header, four bits from the interleaver will be preloaded into the transmitter portion of the modem. These bits are used to pre-compute the DQPSK, and scrambling of the data. These two blocks have a throughput of 1 bit per cycle, but also a latency of four cycles, thus the need to pre-compute four bits. At the same instant the transmitter finishes sending the header, the data path will be activated. The first bit transmitted is the first pre-computed bit, and followed by new readings from then interleaver. The modem reads the data by columns one bit at the time as shown in Figure 19.
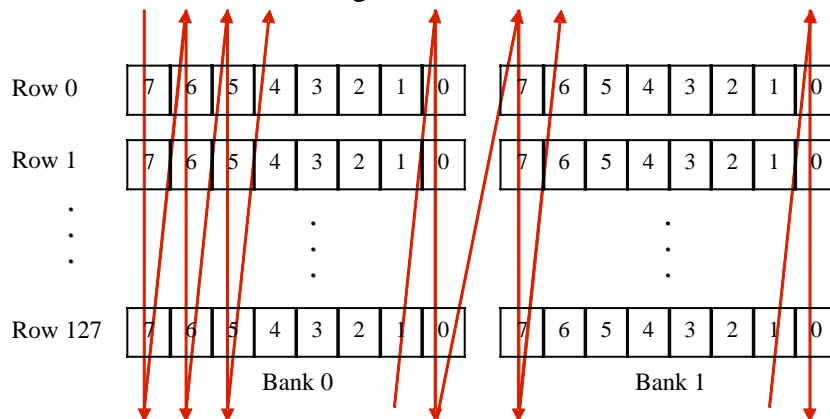
Figure 19: Interleaver column wise bit read operation.

RAM banks are used for the implementation of the interleaver. Each of the blocks has an 8-bit wide data with a depth of 128 addresses. The first bank corresponds to the even Bytes and the second bank to the odd Bytes.

2.3.2.2.2 Parallel to serial unit

The parallel to serial unit converts the Byte format of the processor-modem interface into the serial input the modem requires. This interface is enabled when the interleaver is turned off. The implementation of this unit uses a single RAM bank with 8-bit wide data and 256 addresses in depth. In this case the data is read as it was written, sequentially by rows, one Byte at a time. Each Byte is loaded into a parallel shift register, and once the modem sends the command to read data, the shift register flushes serially bits out until emptied. These operations are continuously performed until all the data has been transmitted.

2.3.2.2.3 Scrambler

Two transmitter scramblers are implemented using the same polynomial ($G(z)=z^{-7}+z^{-4}+1$). In order to completely define the initial state of the data scrambler, two of these units are implemented. The feed through configuration of the scrambler and de-scrambler is self-synchronizing; nevertheless the initial state of the scrambler is set to [001 1011]. Figure 20 shows the scrambler architecture where each delay element (z^{-1}) is implemented with registers.
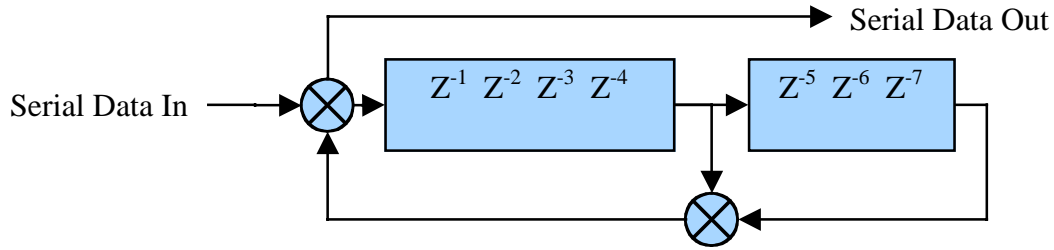
Figure 20: Scrambler Implementation

2.3.2.2.4 PN sequence generator

The structure of the PN sequence generator is the same regardless of the sequence being generated. The sequence generated is a function of the code and initial state being programmed. The code selects the feedback paths, and the initial state is loaded into the delay elements. The output of the generator is a single bit, which changes each consecutive clock cycle, and it is illustrated in Figure 21.
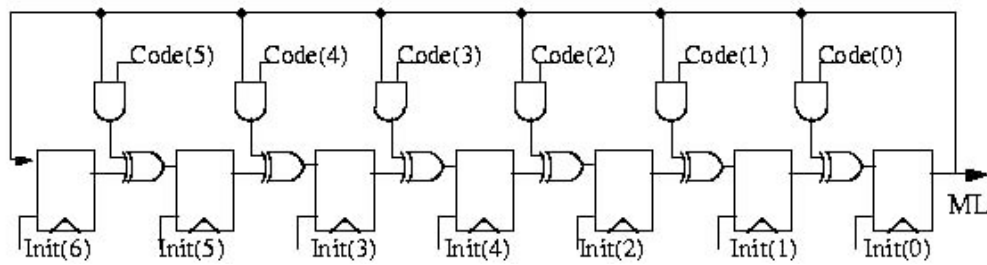
Figure 21: Maximum Length PN sequence generator (15 to 127 chips)

2.3.2.3 Receiver part of the modem

The receiver part of the modem performs the most complex operations. The receiver is divided into two major blocks, acquisition and demodulation. The main purpose of acquisition is to provide code and bit synchronization as well as extraction of the configuration parameters. The demodulation block main purpose is to process the incoming data stream and recover the transmitted data with the extracted configuration parameters.

2.3.2.3.1 Acquisition overview

The acquisition block is divided into three different sequential states; code acquisition, bit synchronization, and configuration data extraction. By processing the preamble, which contains the Sync sequence and the Delimiter, code acquisition and bit synchronization are performed. The incoming data stream is first dispread with the chipping sequence of length 15, decoded with DBPSK, and de-scrambled. Code acquisition provides synchronization between the received spread spectrum signal, and the receiver's local PN sequence. Once code acquisition has finished successfully, the exact phase of the received sequence is known, and data dispreading is possible. In order to reduce overhead, fast acquisition is required. For this purpose dispreading is implemented with matched filters. Bit synchronization is required to determine the starting position of the received bits. In order to dispread correctly the incoming data stream, accurate knowledge of the bit position is required. For this modem data aided bit synchronization is used. The delimiter, known data pattern on the receiver side, is used to flag the starting position of he header.

2.3.2.3.2 Code acquisition

Code acquisition is performed by processing the Sync portion of the preamble. There are two methods of processing an incoming PN sequence to find the correct phase, in parallel or serially. There are benefits and drawbacks to either method, but in regards to energy savings there is no considerable difference on the average.

Serial processing is performed using serial correlators, which are low complexity units with small area implementation cost. On the other hand serial correlators might need to process long packets until the right phase is acquired. Parallel processing is a high complexity block, which requires a large area for implementation. The major benefit for parallel processing is that it only requires a few repetitions of the code in order to find the correct phase.

The architecture selected for code acquisition is a matched filter (MF), which is an implementation of a parallel processing block. The output of the MF is the cross-correlation function between this filter impulse response and the incoming data stream. The filter impulse response is optimized to the transmitted PN sequence. The incoming data stream is a mixture of the transmitted sequence, noise and interferences existing in the channel. The output characteristic of the MF is a flat low energy signal, with a high cross-correlation peak when the desired phase of the sequence is found. Taking into account the major benefit of the MF is the reduction of the overhead, similar energy consumption as the serial counterpart, and the availability of area for implementation this option is the optimum for this application.
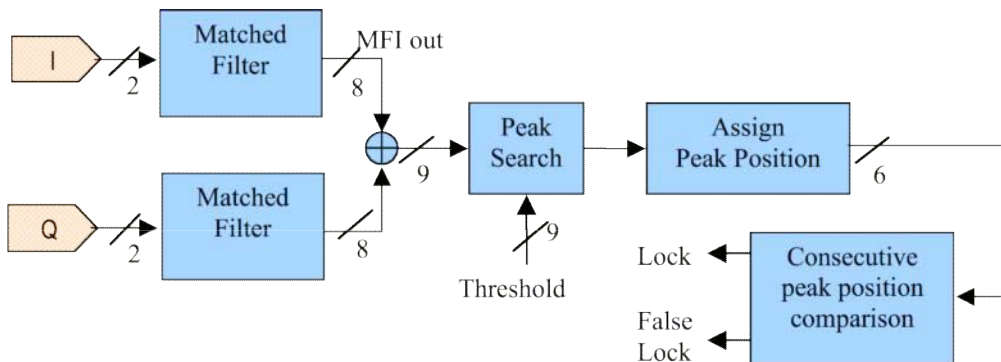
Figure 22: Code Acquisition Portion of the Acquisition Block

The inputs for the code acquisition block are I and Q signals represented by 2 bits, 4 times over sampled (4.27Msps), with two's complement representation. The MF output is represented by a two's complement 8 bit wide signal. Both I and Q MF output energy signals are then combined and used to search for correlation peaks. The search of a correlation peak is implemented by comparing the output energy of the matched filter to a predetermined threshold. Once a peak has been detected, a position is assigned to this peak. The code acquisition block will search for the next peak at the same position (plus or minus one) as the first peak. The peak position might vary slightly due to the presence of noise in the channel, thus the peak search is expanded from the first peak position ±1.

The position tracking is implemented by a modulo 60 counter. The acquisition PN sequence has a length of 15 chips/bit, and the receiver takes 4 samples/chip. Thus the total number of samples in one entire PN sequence repetitions is 60. This counter is clocked at the sampling rate (4.27MHz). Three consecutive peaks are necessary in order to declare code acquisition, which is signaled by the high state of the lock signal. If three consecutive peaks are not found, false lock is declared. When false lock occurs the code acquisition is reset and returns to the initial peak search state. The code acquisition algorithm is depicted in Figure 23.
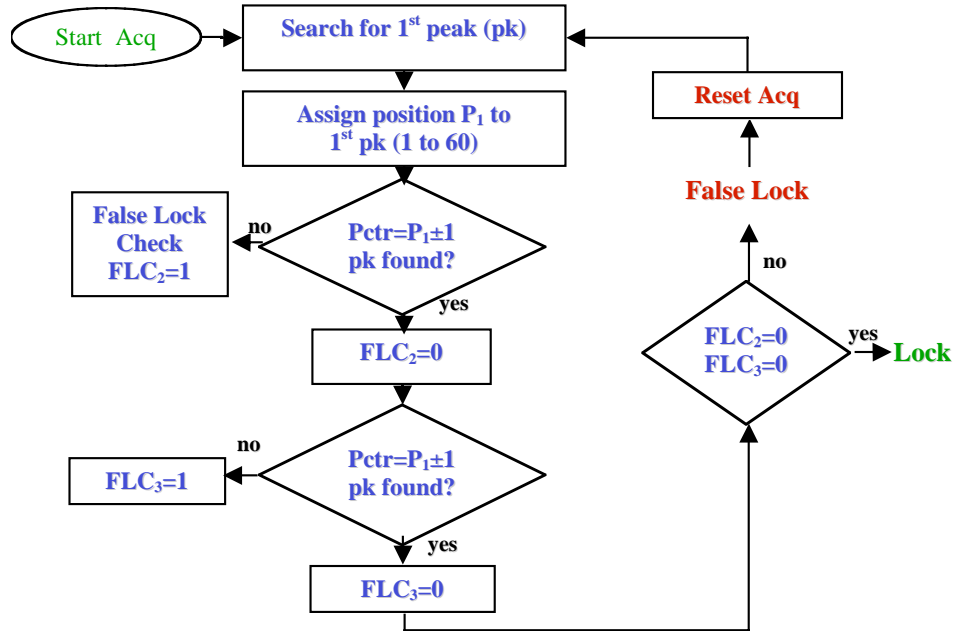
Figure 23: Code Acquisition Algorithm

As soon as the code acquisition lock has been declared, the acquisition PN sequence generator is enabled. The resulting PN sequence is in-phase with the received I and Q sequences, thus allowing data demodulation. The MF and code acquisition logic are placed in disable mode, and serial correlators are enabled to process the incoming data for dispreading replacing the MF.

2.3.2.3.3 Bit synchronization

Bit synchronization starts by enabling the I and Q serial correlators. These correlators take the two bit wide, two's complement input samples, and the PN sequence to dispread the data. The dispread energy signal is an 8-bit value, which is first truncated to 1 bit by making a hard decision. This bit is then sent for decoding to the DBPSK unit, which outputs a bit to be de-

scrambled. The inputs to the serial correlator arrive at the sampling rate (4.27Msps), and the outputs are processed at the bit rate for the preamble and header (66.67Ksps).
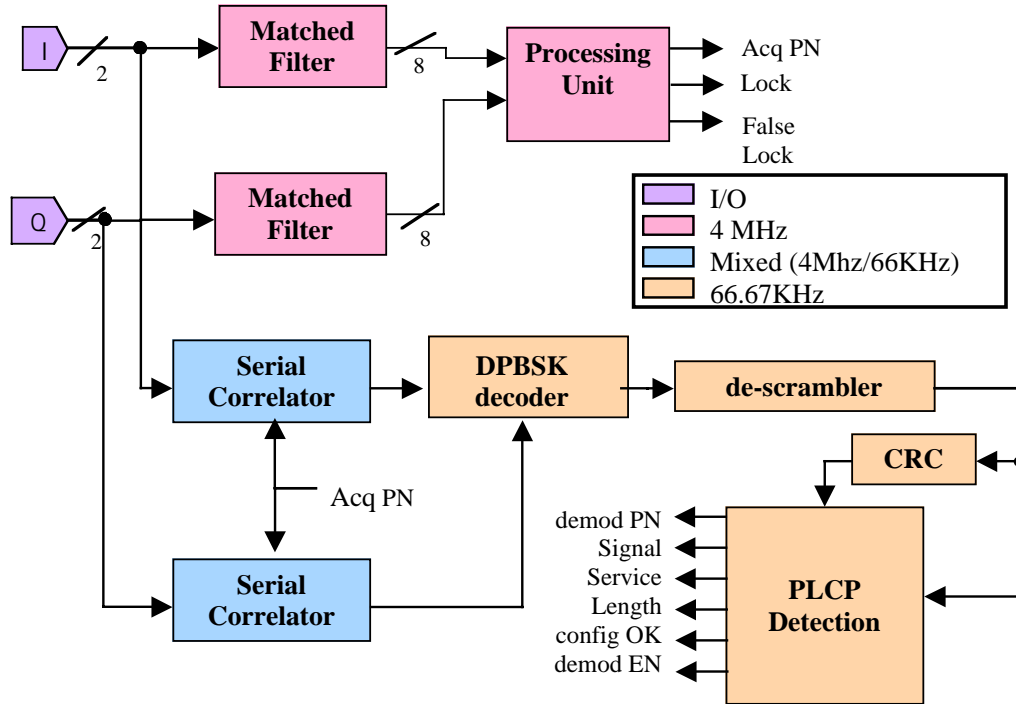
Figure 24: Bit Synchronization and Code Acquisition Blocks

The acquisition block performs bit synchronization by searching for the pre-assigned delimiter pattern [1111 0011 1010 0000]. The search is performed on the dispread, DBPSK decoded, and de-scrambled received bits. All received bits are entered into a 16 bit wide FIFO, which compares each single bit to the delimited pattern. Once all the bits have been positively matched bit synchronization is declared, and the delimiter search state commences.

2.3.2.3.4 Receiver configuration

As soon as the delimiter has been found, the following incoming bits are stored in the header FIFO. The first 8 bits are assigned to the signal field, the next following 8 bits to the service field and the final 16 bits to the length field. The CRC block is enabled as soon as bit synchronization has been achieved, and all bits entering the header FIFO are also sent to the CRC block for processing. Once the signal, service and length fields have been received, the receiver part of the modem is configured with those parameters. At the same instant the last bit of the CRC code is received, the data demodulation block is enabled. At this point the configuration parameters correctness are uncertain. The CRC needs one more clock cycle (at the header data rate 66.67KHz) to finish the computation and validate the header. In the mean time the payload data is received and demodulated. Once the CRC has finished the computation two lines of action are possible. If the CRC check is positive, configuration OK signal switches to high state and the demodulation process keeps active. If the CRC check is negative, the demodulator is disabled, false acquisition is declared, and the receiver modem switches to the beginning of the code acquisition state.

2.3.2.3.5 Data demodulation

Data demodulation starts right after the last bit of the header CRC has been received. Since the demodulation process is programmable, the data path varies according to the configuration parameters selected as shown in section 2.3.1.2.2. The demodulator first takes the two-bit representation of I and Q and dispreads the signal with a serial correlator. The output of the correlator is a 10-bit wide signal, which first is truncated to a one-bit signal by performing a hard decision. The dispread bit is decoded with either DBPSK, or DQPSK decoder. The decoded bits are sent to the de-scrambler and/or the de-interleaver, as selected by the header, in order to finish the demodulation process and recover the original transmitted data.
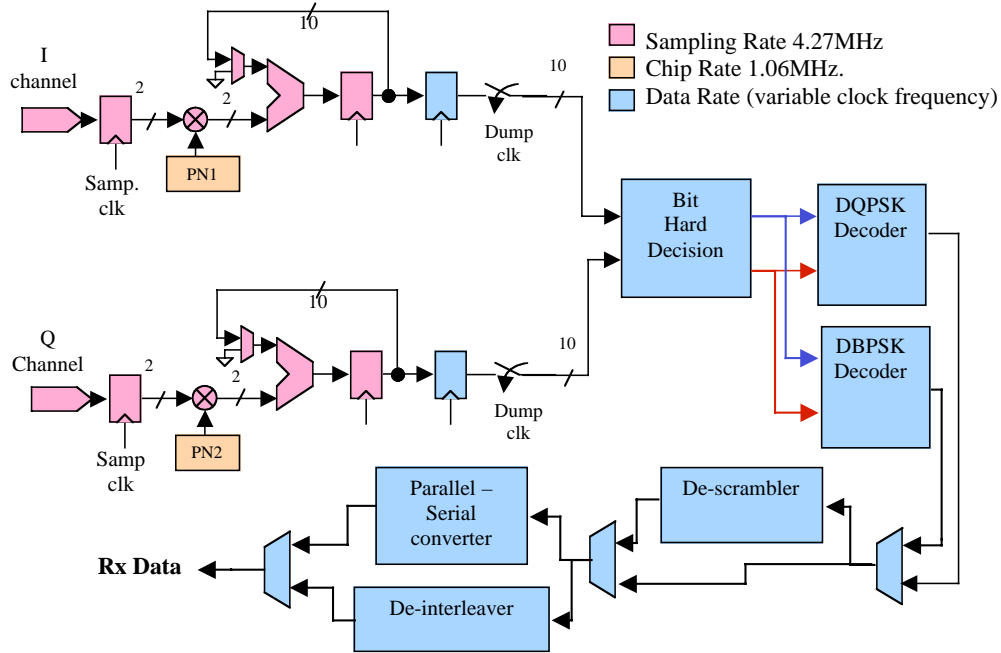
Figure 25: Demodulation Architecture

The first portion of the demodulator computes the incoming samples at the sampling rate (4.27MHz). Once the incoming sequence has been dispread, the decoders, de-scrambler, de-interleaver and parallel-to-serial blocks process the bits at the data rate. This processing frequency is a variable rate according to Table 6.

2.3.2.3.6 De-scrambler

The de-scrambler is used when b1 of the service field is set to '1'. Figure 26 shows the implementation for the used de-scrambler. When the de-scrambler is enabled, all the payload bits are sent to this block. The initial state of the block is [110 1100], same as the transmitter counter part. The implementation of the delay elements (z^{-n}) is registers.
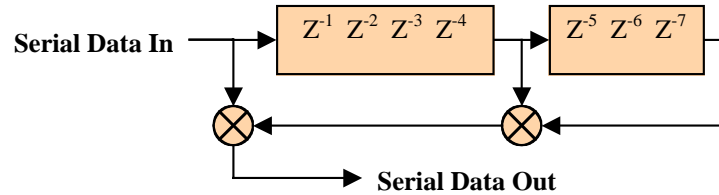
Figure 26: De-scrambler architecture.

When b1 of the service field is set to '0', the de-scrambler is not being used and the delay elements are disabled.

2.3.2.3.7 De-interleaver

The de-interleaver works in combination with the interleaver used in the transmitter side and explained in section 2.3.2.2.1. The implementation for de-interleaver block is composed of 16 RAM banks of 1-bit width and depth 128 addresses, similar to the RAM block depicted in Figure 29. Figure 27 shows a simplified RAM picture to illustrate the write process.
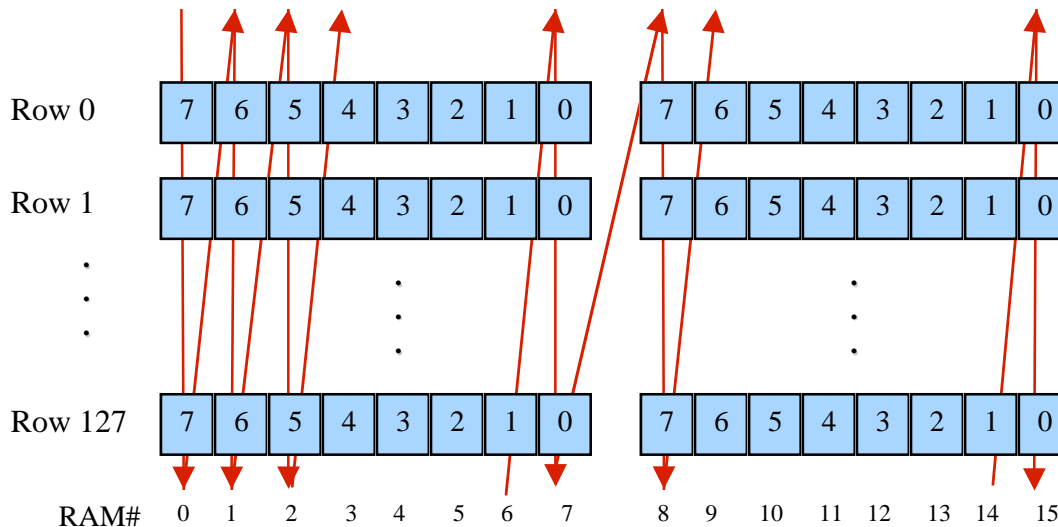
Figure 27: De-Interleaver Write Operation

The write process takes the serial input data from the de-scrambler block. The address controller computes address changes depending on the length of the payload. For example if the payload has a length of 4 Bytes only address 0 and 1 will be used for de-interleaving. Received bits are written into the same RAM block until the maximum address has been reached, 1 in the previous example. At that point the next incoming bit is sent to the next RAM block address 0. The operation is repeated until all payload data has been written to the RAM blocks.
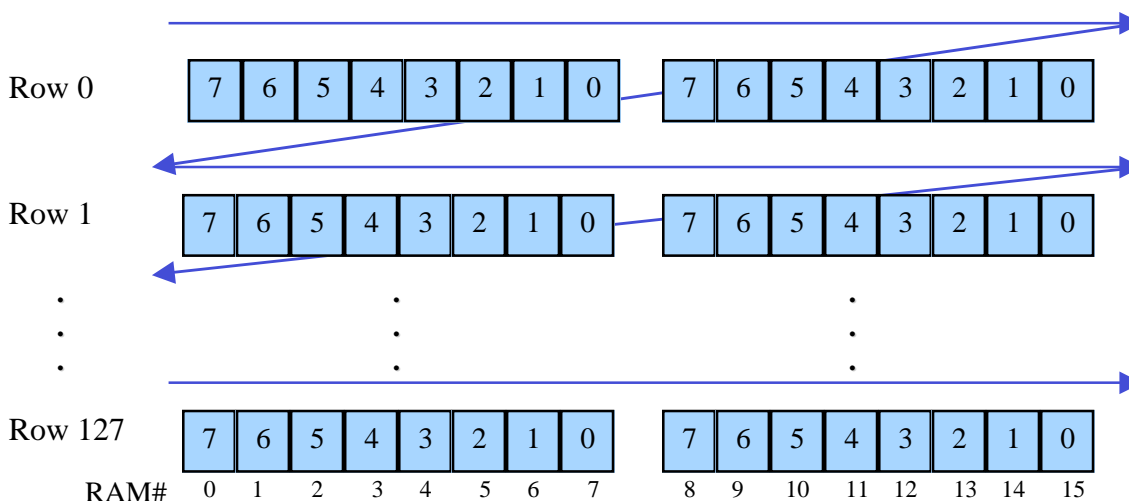
Figure 28: De-Interleaver Read Operation.

23

Once all received bits have been written the modem signals the processor (section 2.3.4), the receiver RAM (FIFO) is ready to be read. The read operation is performed at the processor's command. Read data is transferred in a Byte format. The RAM is read by rows, transferring 2 Bytes per row until depleted. The read operation is depicted in Figure 28.

2.3.2.3.8 Serial-to-parallel converter

The function of the serial-to-parallel converter is simply to transform the received data into the parallel Byte wide format for the processor. The implementation for this block is similar as for the de-interleaver. In this case 8 1-bit wide RAM blocks with 256 addresses in parallel are used.
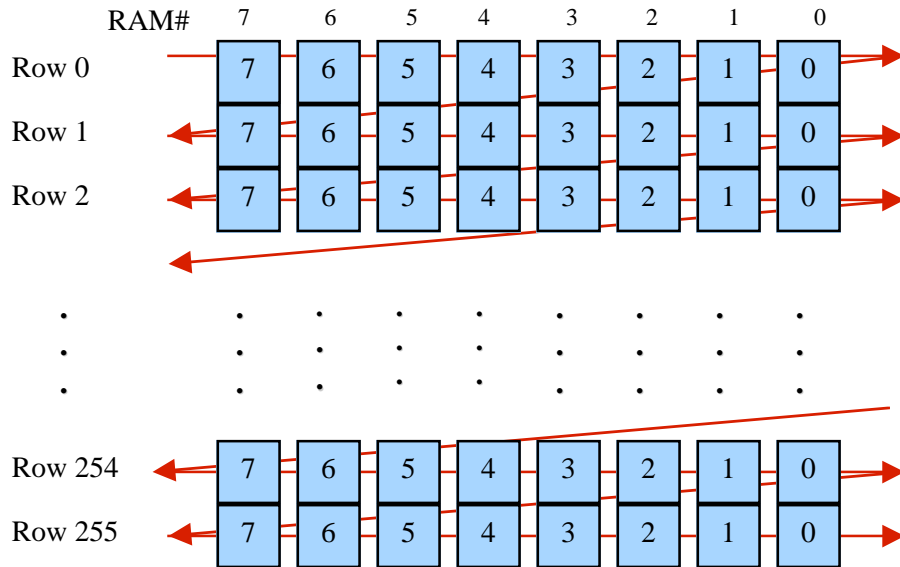
Figure 29: Serial-to-Parallel write/read RAM process.

The write process takes the serial input data from the de-scrambler block. A bit is written each time until the row is completely full. Then the address increases by one. The same write operation is repeated until all incoming data has been written into the RAM banks. This operation is shown in Figure 29. The read operation is performed at the processor command. Data is read one byte at the time. Each Byte is composed with the data contained in all 8 RAM banks with the same address location.

2.3.2.4 Clock management

The clock management block generates and distributes all clocks throughout the modem. This block uses several digital delay lock loops (DLL) units available in the VirtexE devices. The input clock is a stable 25.6MHz clock, which then is divided and multiplied to provide the modem with a sampling (system) clock of 4.27MHz. Further dividing the system clock by four we obtain the chipping clock of 1.067MHz. Each DLL provides limited resources of clock frequency multiplication or division. In order to obtain the desired frequencies the clock network tree depicted in Figure 30 was developed.

The input clock frequency first is multiplied by 2 to increase the frequency to 51.2MHz. This clock frequency is divided by 1.5 in order to yield a 34.13MHz; which is further divided by 8 to generate the sampling clock with a frequency of 4.267MHz.
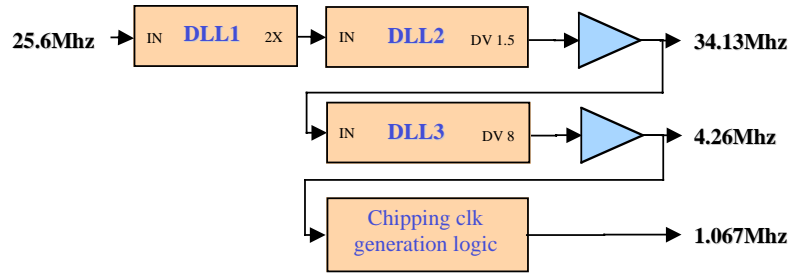
Figure 30: Clock Generation Tree

All DLL block outputs drive a clock buffer with dedicated clock lines. These lines are low skew and low delay. DLL blocks have the disadvantage of not being able to be dynamically re-configurable. The receiver chipping clock phase must be adjusted for each acquisition. This DLL shortcoming means the chipping clock must be manually generated. A manually generated clock does not have access to the low-skew lines, thus making this clock line vulnerable to high clock skews and endangering data reliability. In order to solve this problem, a single clock line (system clock) of 4.27MHz is used throughout the FPGA. The implementation of a slower processing rate block is as follows. The clock line uses the system clock. The slower rate is controlled by using an enable signal, which allows the circuitry to process data using the system clock only during desired periods of time.

2.3.3 Differential encoding and decoding

2.3.3.1 Rationale for using differential PSK

Phase-shift keying (PSK) has a lot of advantages. Potentially, binary PSK (BPSK) is the most noise resistant type of two-position signals because it comprises antipodal signals, which have the maximum Euclidean distance between them. For example, BPSK requires twice lower transmitter power than orthogonal binary frequency-shift keying (BFSK) for achieving the same bit error rate (BER). Quaternary PSK (QPSK) is a set of 4 bi-orthogonal signals. Potentially, it provides twice higher channel throughput than orthogonal BPSK with the same transmitter bandwidth. In addition, PSK is invariant to the signal amplitude. This is especially important in the channels with fading and multipath propagation. Along with these advantages, PSK has an inherent problem caused by the initial phase ambiguity. Differential encoding and decoding allows this problem to be overcome. Differential encoding/decoding in is not the only way to overcome the problem of the phase ambiguity. However, it is the most effective and efficient way in the majority of practical situations. Various types of phase-difference modulation (PDM), especially second order PDM, are also very useful in Doppler channels. Since differential encoding and decoding are utilized in the HDR-01 modem, their detailed analysis is provided in the subsequent sections of this chapter.

2.3.3.2 Differential BPSK

2.3.3.2.1 Differential encoding of BPSK

The sequence of symbols intended for transmission is:

$$a_1, a_2, a_3, \ldots, a_k, \ldots$$

25

where k is a positive integer. Differential encoding is intended to cope with phase ambiguity, i.e. possibility of phase inversion. Differential encoding transforms sequence $\{a_k\}$ into sequence $\{b_k\}$ according to the following rule:

$$b_1 = a_1, \quad b_2 = a_2 \oplus b_1, \quad b_3 = a_3 \oplus b_2, \ldots$$

In general case:

$$b_1 = a_1, \text{ and } b_k = a_k \oplus b_{k-1} \text{ for } k \geq 2$$

The differential encoding procedure for BPSK can be performed by the structure whose block diagram is shown in Figure 31. The output sequence of the differential encoder $\{b_k\}$ can be fed into the modulator immediately or after further processing (for example spectrum spreading).
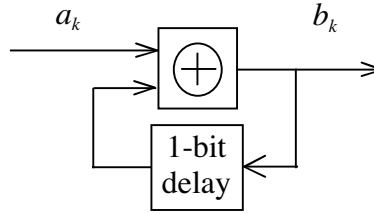
Figure 31: Block Diagram of the Differential Encoder for BPSK

2.3.3.2.2 Differential decoding of BPSK
The sequence being received is as follows:

$$b_1*, b_2*, b_3*, \ldots, b_k*, \ldots$$

Sequence b_k* reflects the decisions made in the demodulator. Because of noise and interference, b_k* does not always coincide with b_k. Differential decoding transforms sequence $\{b_k*\}$ into sequence $\{a_k*\}$ according to the following rule:

$$a_1* = b_1*, \quad a_2* = b_2* \oplus b_1*, \quad a_3* = b_3* \oplus b_2*, \ldots$$

In general case:

$$a_k* = b_k* \oplus b_{k-1}*$$

The differential decoding procedure for BPSK can be performed by the structure whose block diagram is shown in Figure 32.
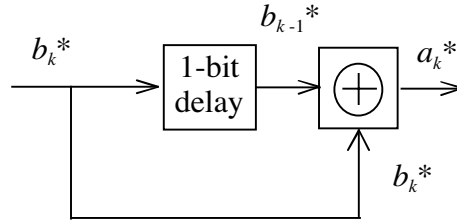
Figure 32: Block Diagram of the Differential Decoder for BPSK

2.3.3.2.3 Examples of BPSK differential encoding and decoding
Two examples of BPSK differential encoding and decoding are shown in Table 7 and Table 8. Table 7 illustrates the case of the ideal transmission (i.e. no error and no immediate change of the phase occurs in the system).

No.	1	2	3	4	5	6	7	8	9	10	11	12
a_k	0	1	1	0	1	1	1	0	0	0	0	1
b_k	0	1	0	0	1	0	1	1	1	1	1	0
$b_k{}^*$	0	1	0	0	1	0	1	1	1	1	1	0
$a_k{}^*$	0	1	1	0	1	1	1	0	0	0	0	1

Table 7: Ideal BPSK Transmission Case

Table 8 illustrates the case of the transmission with one error in sequence $\{b_k{}^*\}$ (symbol $b_3{}^*$ has been demodulated incorrectly) and one immediate 180^0 change of the phase between symbols $b_7{}^*$ and $b_8{}^*$. All the errors in the table are underlined.

No.	1	2	3	4	5	6	7	8	9	10	11	12
a_k	0	1	1	0	1	1	1	0	0	0	0	1
b_k	0	1	0	0	1	0	1	1	1	1	1	0
$b_k{}^*$	0	1	_1_	0	1	0	1	_0_	_0_	_0_	_0_	_1_
$a_k{}^*$	0	1	_0_	_1_	1	1	1	_1_	0	0	0	1

Table 8: Single Error BPSK Transmission Case

As shown in Table 8, one error in sequence $\{b_k{}^*\}$ (see symbol $b_3{}^*$) causes two errors in sequence $\{a_k{}^*\}$ (see symbols $a_3{}^*$ and $a_4{}^*$). Thus, differential encoding and decoding double single errors. On the other hand, the immediate 180^0 change (inversion) of the signal phase in sequence $\{b_k{}^*\}$ (see symbols $b_8{}^*$, $b_9{}^*$, $b_{10}{}^*$, $b_{11}{}^*$, and $b_{12}{}^*$) causes only one error in sequence $\{a_k{}^*\}$ (see symbol $a_8{}^*$) at the output of the decoder. In principle, this error can occur in the bit during which inversion takes place or the bit just after inversion. The case of N errors in sequence $\{b_k{}^*\}$ following one after another is illustrated in Table 9.

No.	1	2	3	4	5	6	7	8	9	10	11	12
a_k	0	1	1	0	1	1	1	0	0	0	0	1
b_k	0	1	0	0	1	0	1	1	1	1	1	0
$b_k{}^*$	0	1	0	_1_	_0_	1	_0_	_0_	_0_	1	1	0
$a_k{}^*$	0	1	1	_1_	1	1	1	0	0	_1_	0	1

Table 9: N Errors BPSK Transmission Case

It is easy to notice that when there is an error burst of N errors in sequence $\{b_k{}^*\}$ following one after another, there will be only two errors in sequence $\{a_k{}^*\}$: one error at the beginning of the burst and another one immediately after the end of the burst. Due to the doubling of single errors in the differentially encoded sequence, the employment of codes developed for correction of doubled errors or interleaving can be useful in the considered case.

2.3.3.3 Differential QPSK

2.3.3.3.1 Differential encoding of QPSK
The sequence of symbols intended for transmission is as follows:

$$a_1, a_2, a_3, \ldots, a_k, \ldots$$

where k is a positive integer. This sequence should be first converted into two parallel sequences:

$$i_1 = a_1, i_2 = a_3, i_3 = a_5, \ldots, i_k = a_{2k-1}, \ldots$$

$$q_1 = a_2, q_2 = a_4, q_3 = a_6, \ldots, q_k = a_{2k}, \ldots$$

Thus, sequence $\{i_k\}$ contains all odd bits of sequence $\{a_k\}$, and sequence $\{q_k\}$ contains all even bits of sequence $\{a_k\}$. Differential encoding transforms sequences $\{i_k\}$ and $\{q_k\}$ into sequences $\{I_k\}$ and $\{Q_k\}$ according to the following rule:

$$I_k = \overline{(i_k \oplus q_k)} \cdot (i_k \oplus I_{k-1}) + (i_k \oplus q_k) \cdot (q_k \oplus Q_{k-1})$$

$$Q_k = \overline{(i_k \oplus q_k)} \cdot (q_k \oplus Q_{k-1}) + (i_k \oplus q_k) \cdot (i_k \oplus I_{k-1})$$

Since $I_{k-1} = 0$ and $Q_{k-1} = 0$ when $k = 1$, it follows that:

$$I_1 = \overline{(i_1 \oplus q_1)} \cdot i_1 + (i_1 \oplus q_1) \cdot q_1$$

$$Q_1 = \overline{(i_1 \oplus q_1)} \cdot q_1 + (i_1 \oplus q_1) \cdot i_1$$

The differential encoding procedure for QPSK can be performed by the structure whose block diagram is shown in Figure 33. Here, sequences $\{i_k\}$ and $\{q_k\}$ are formed from the sequence $\{a_k\}$ first. Then, sequences $\{i_k\}$ and $\{q_k\}$ are converted into sequences $\{I_k\}$ and $\{Q_k\}$ according to the equations above. Sequences $\{I_k\}$ and $\{Q_k\}$ from the output of the differential encoder can be fed into the modulator immediately or after further processing (for example spectrum spreading). In HDR-1 modem, these sequences from the output of the differential encoder are fed into the spread spectrum modulator.

Although differential encoders for both BPSK and QPSK require one-symbol delay elements, there is significant difference between them because one symbol corresponds to one bit in BPSK and two bits in QPSK (compare Figure 31 and Figure 33).
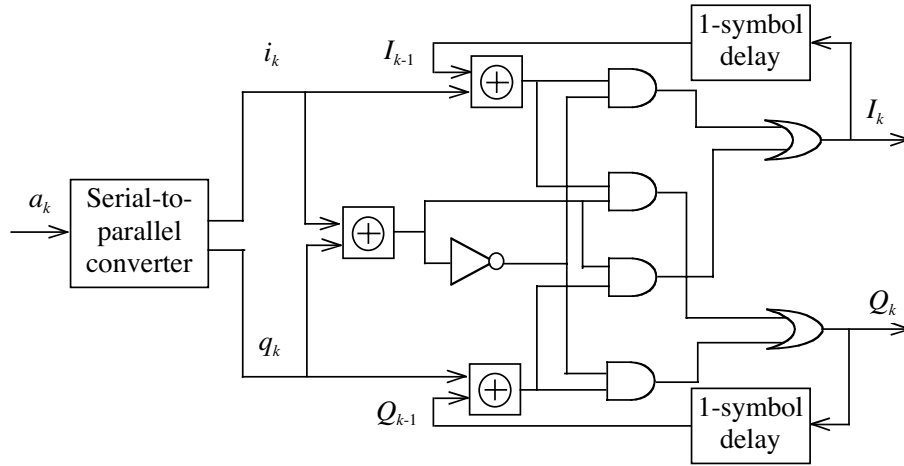
Figure 33: Block diagram of the differential encoder for QPSK

The structure of a differential encoder for BPSK can be obtained as a special case of a differential encoder for QPSK when input signal is the same for I and Q components.

2.3.3.3.2 *Differential decoding of QPSK*

The sequences of symbols being received in the channels of in-phase and quadrature components are as follows:

$$I_1^*, I_2^*, I_3^*, \ldots, I_k^*, \ldots$$

$$Q_1^*, Q_2^*, Q_3^*, \ldots, Q_k^*, \ldots$$

28

Sequences $\{I_k^*\}$ and $\{Q_k^*\}$ reflect the decisions made in the channels of in-phase and quadrature components. Since noise and interference can cause errors, I_k^* does not always coincide with I_k, and Q_k^* does not always coincide with Q_k. Differential decoding transforms sequences $\{I_k^*\}$ and $\{Q_k^*\}$ into sequence $\{i_k^*\}$ and $\{q_k^*\}$ according to the following rules:

$$i_k^* = \overline{(I_k^* \oplus Q_k^*)} \cdot (I_k^* \oplus I_{k-1}^*) + (I_k^* \oplus Q_k^*) \cdot (Q_k^* \oplus Q_{k-1}^*)$$

$$q_k^* = \overline{(I_k^* \oplus Q_k^*)} \cdot (Q_k^* \oplus Q_k^*) + (I_k^* \oplus Q_k^*) \cdot (I_k^* \oplus I_k^*)$$

Since $I_{k-1}^* = 0$ and $Q_{k-1}^* = 0$ when $k = 1$, it follows from (4.7) and (4.8) that

$$i_1^* = \overline{(I_1^* \oplus Q_1^*)} \cdot I_1^* + (I_1^* \oplus Q_1^*) \cdot Q_1^*$$

$$q_1^* = \overline{(I_1^* \oplus Q_1^*)} \cdot Q_1^* + (I_1^* \oplus Q_1^*) \cdot I_k^*$$

The differential decoding procedure for QPSK can be performed by the structure whose block diagram is shown in Figure 34.
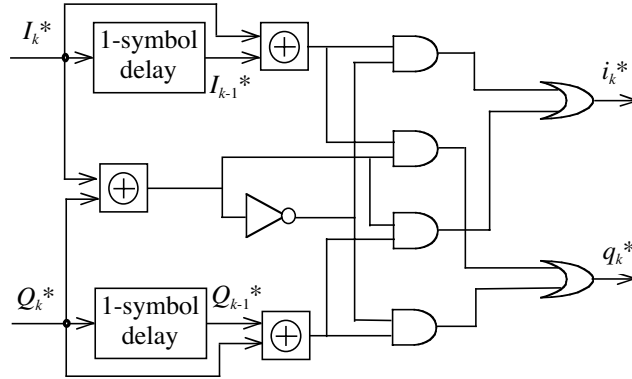
Figure 34: Block diagram of the differential decoder for QPSK

In the simplest version of communication system, sequences $\{I_k^*\}$ and $\{Q_k^*\}$ come from the demodulator, and sequences $\{i_k\}$ and $\{q_k\}$ enter the de-interleaver. Like in the case of differential encoders, there is significant difference between the one-symbol delay elements required for differential decoders for BPSK and QPSK because one symbol corresponds to one bit in BPSK and two bits in QPSK (compare Figure 32 and Figure 34). It is also easy to show that the structure of a differential decoder for BPSK can be obtained as a special case of a differential decoder for QPSK when input signal is the same for I and Q components.

2.3.3.3.3 Examples of QPSK Differential Encoding and Decoding
An example of QPSK differential encoding is shown in Table 10.

Table 10: QPSK Differential Encoding

a_k	0	1	1	0	1	1	1	0	0	0	0	1	1	0	1	1	1	0
i_k	0		1		1		1		0		0		1		1		1	
q_k	1		0		1		0		0		1		0		1		0	
I_k	1		0		1		1		1		1		1		0		1	
Q_k	0		0		1		0		0		1		0		1		1	

29

An example of QPSK differential decoding for the case of the ideal transmission (i.e. no error and no immediate change of the phase occurs in the system) is shown in Table 11.

Table 11: QPSK Differential Encoding Ideal Case

I_k^*	1	0	1	1	1	1	1	0	1
Q_k^*	0	0	1	0	0	1	0	1	1
i_k^*	0	1	1	1	0	0	1	1	1
q_k^*	1	0	1	0	0	1	0	1	0

Table 12, Table 13 and Table 14 illustrate the various cases of the transmission with errors in sequences $\{I_k^*\}$ and $\{Q_k^*\}$ (the errors in the tables are underlined). In Table 12, the first error occurs in symbol I_2^*, Q_2^*. This symbol should be 0, 0. However, it has been demodulated as 1, 0. As a result of this error, two symbols (i_2^*, q_2^* and i_3^*, q_3^*) have been distorted at the output of the QPSK differential decoder. Symbol i_2^*, q_2^* (which should be 1, 0) has been transformed into 0, 0; and symbol i_3^*, q_3^* (which should be 1, 1) has been transformed into 0, 1. The second error in Table 12 occurs in symbol I_5^*, Q_5^*. This symbol should be demodulated as 0, 0. However, it has been demodulated as 1, 0. As a result of this error, symbols i_5^*, q_5^* and i_6^*, q_6^* have been distorted at the output of the QPSK differential decoder. Symbol i_5^*, q_5^* (which should be 0, 0) has been transformed into 0, 0; and symbol i_6^*, q_6^* (which should be 0, 1) has been transformed into 0, 1.

Table 12: QPSK Differential Encoding Error Case

I_k^*	1	<u>1</u>	1	1	1	1	1	0	1
Q_k^*	0	0	1	0	<u>1</u>	1	0	1	1
i_k^*	0	<u>0</u>	<u>0</u>	1	0	0	1	1	1
q_k^*	1	0	1	0	<u>1</u>	<u>0</u>	0	1	0

In Table 13, the 1st, 4th, and 7th symbols have been demodulated incorrectly. The 1st symbol (I_1^*, Q_1^*) has been demodulated as 0, 0 instead of 1, 0. The 4th symbol (I_4^*, Q_4^*) has also been demodulated as 0, 0 instead of 1, 0. The 7th symbol (I_7^*, Q_7^*) has been demodulated as 0, 1 instead of 1, 0. As a result of each of these errors, two symbols have been distorted at the output of the QPSK differential decoder. The 1st error causes the transformation of i_1^*, q_1^* from 0, 1 into 0, 0 and transformation of i_2^*, q_2^* from 1, 0 into 0, 0. The 2nd error causes the transformation of i_4^*, q_4^* from 1, 0 into 1, 1 and transformation of i_5^*, q_5^* from 0, 0 into 0, 1. The 3rd error causes the transformation of i_7^*, q_7^* from 1, 0 into 0, 1 and transformation of i_8^*, q_8^* from 1, 1 into 0, 0.

Table 13: QPSK Differential Encoding Demodulation Error Case

I_k^*	<u>0</u>	0	1	<u>0</u>	1	1	<u>0</u>	0	1
Q_k^*	0	0	1	0	0	1	<u>1</u>	1	1
i_k^*	0	<u>0</u>	1	1	0	0	<u>0</u>	<u>0</u>	1
q_k^*	<u>0</u>	0	1	<u>1</u>	<u>1</u>	1	<u>1</u>	<u>0</u>	0

It follows from Table 12 and Table 13 that all types of single errors in sequences $\{I_k^*\}$ and $\{Q_k^*\}$ cause two errors in sequences $\{i_k^*\}$ and $\{q_k^*\}$. Thus, differential encoding and decoding QPSK double single errors similarly to the case of BPSK.

Table 14 illustrates the case when N errors follow one after another in sequences $\{I_k*\}$ and $\{Q_k*\}$. Here, we can see two such error bursts. The first of them consists of the first three symbols of sequences $\{I_k*\}$ and $\{Q_k*\}$. The second error burst consists of the 6th and 7th symbols of sequences $\{I_k*\}$ and $\{Q_k*\}$.

Table 14: QPSK Differential Encoding N-Errors Case

I_k*	0	1	0	1	1	1	0	0	1
Q_k*	0	0	1	0	0	0	1	1	1
i_k*	0	0	1	1	0	0	1	0	1
q_k*	0	1	1	1	0	0	1	0	0

As shown in Table 14, N errors following one after another in sequences $\{I_k*\}$ and $\{Q_k*\}$ cause the error burst in sequences $\{i_k*\}$ and $\{q_k*\}$. The length of the error burst is $N + 1$ and its structure depends on types of the errors in sequences $\{I_k*\}$ and $\{Q_k*\}$.

Table 15 illustrates the case of one immediate 180^0 change (inversion) of the phase between symbols I_4*, Q_4* and I_5*, Q_5*.

Table 15: QPSK 180-Degree Phase Change (Inversion) Case

I_k*	1	0	1	1	0	0	0	1	0
Q_k*	0	0	1	0	1	0	1	0	0
i_k*	0	1	1	1	1	0	1	1	1
q_k*	1	0	1	0	1	1	0	1	0

As follows from Table 15, the immediate 180^0 change of the signal phase in sequences $\{I_k*\}$ and $\{Q_k*\}$ causes only one error in sequences $\{i_k*\}$ and $\{q_k*\}$. In the case shown in Table 15, this error occurs in the symbol i_5*, q_5*. In principle, this error can occur in the symbol during which inversion takes place or the symbol just after inversion. Due to the doubling of single errors in the differentially encoded QPSK, the employment of codes developed for correction of doubled errors or interleaving can increase the reliability of DQPSK transmission.

2.3.4 Network processor interface

The network processor – modem interface is a memory mapped, with the following signals 8 bits for data command transfer, chip select (CS#), output enable (OE#), write enable (WE#), write/read select (RD#WR), address line (A04), ready (RDY), net to modem interrupt (NTM_IRPT), modem to net interrupt (MTN_IRPT).

There are four available commands, transmission enable, modem reset, configuration (only for the transmitter) and data transfer. A transmission must always start by configuring the modem, followed by loading the transmit data into the TX FIFO. After this the modem is ready for transmission. The receiver only needs a command to be placed into receive mode.

2.3.4.1 Command description

A processor to modem transfer starts by first sending an interrupt pulse (NTM_IRPT). The enable signals follow the data transfer protocol described in section 10.5.5 (variable latency I/O interface) of Intel SA-1110 microprocessor developer's manual. The data lines are used for command as well as data transfer. After the interrupt pulse the processor first sends the command using the signals according to the SA-1110 manual.

Table 16: Processor-modem Interface Commands.

Enable Transmission	Configuration	Reset	Data Transfer
X'01	X'02	X'04	X'08

The two MSB bits of the command word corresponds to the address of the transmitter (X'40) or the receiver (X'80). The final command word is composed of the portion of the modem the command is directed to and the actual command described in Table 16. For example to configure the transmitter modem send X'42 as the command word.

The configuration of the transmitter modem is performed by first sending the command and then followed by four Bytes. These Bytes correspond to the signal, service and length (2 Bytes) fields. Data transfer to the modem is performed first sending the data transfer command, and followed by data Bytes. The number of Bytes has to be the same as in the length field plus 1 (e.g. length =5, number of data Bytes=6). Once the receiver modem has finished processing all the data and placed it in the RX FIFO, it will signal the processor that this data is available for retrieval. At this point the modem sends an interrupt (MTN_IRPT) pulse. Once the processor is ready to retrieve all the data, it will send a command for data transfer (X'88). Once the command is received the modem will read the RX FIFO and place all the data into the data bus following the SA-1110 protocols.

2.3.5 Code development and simulation

This modem was developed mainly using VHDL language. All simulations have been performed using activeHDL from Aldec, synthesis with FPGA compiler II from Synopsys, and place and route with Xilinx ISE 4.2. Code development has been done from the bottom up, starting with basic blocks and building to implement the major blocks. The code has been written in a modular block style. This allows modifying or replacing blocks when changes are necessary. Several basic building blocks have been implemented using cores from Xilinx LogiCore, which are area and speed optimized for the specific targeted FPGA. Table 17 enumerates all the files used for the modem. The indexing represents the hierarchy in the modem. VHDL files have a .vhd extension and the LogiCore generated files have a netlist file extension .edn.

Table 17: FPGA Modem Design Files

```
PACC_modem_if.vhd                          (Top level entity. It contains all FPGA interfaces)

      PACC_clk_management.vhd              (clock generation block)
            CLKDLL, CLKBUF                              (basic VirtexE blocks).
      PACC_selftest_ctrl.vhd               (self test unit)
      PACC_interleaver.vhd                 (interleaver block)
            Interleaver_RAM.edn                         (interleaver RAM basic block)
            Parallel_to_serial_RAM.edn            (parallel to serial RAM block)
      PACC_deInterleaver.vhd               (de-interleaver unit)
            De_interleaver_RAM.edn                 (de-interleaver RAM basic block)
            Parallel_to_serial_RAM.edn            (parallel to serial RAM block)
      Processor_interface.vhd              (modem-processor interface logic)

      PACC_modem_core.vhd                   (Modem block)
            PACC_acq.vhd                 (Acquisition block)
                  PACC_mf.vhd                (Acquisition Matched Filter block)
                        MFcodegen.vhd                  (MF code generation block)
                        mf_add_sub_in2_2_out2.edn  (2 bit input MF tap)
                        mf_add_sub_in2_2_out3.edn  (2-bit, 3-bit output MF tap)
                        mf_add_sub_in2_3_out3.edn  (2-bit, 3-bit inputs, 3-bit output tap)
                        mf_add_sub_in2_3_out4.edn  (2-bit, 3-bit inputs, 4-bit output tap)
```

```
                    mf_add_sub_in2_4_out4.edn  (2-bit, 4-bit inputs, 4-bit output tap)
                    mf_add_sub_in2_4_out5.edn  (2-bit, 4-bit inputs, 5-bit output tap)
                    mf_add_sub_in2_5_out5.edn  (2-bit, 5-bit inputs, 5-bit output tap)
                    mf_add_sub_in2_5_out6.edn  (2-bit, 5-bit inputs, 6-bit output tap)
                    mf_add_sub_in2_6_out6.edn  (2-bit, 6-bit inputs, 6-bit output tap)
                    mf_add_sub_in2_6_out7.edn  (2-bit, 6-bit inputs, 7-bit output tap)
                    mf_add_sub_in2_7_out7.edn  (2-bit, 7-bit inputs, 7-bit output tap)
                    mf_add_sub_in2_7_out8.edn  (2-bit, 7-bit inputs, 8-bit output tap)
                    mf_add_sub_in2_8_out8.edn  (2-bit, 8-bit inputs, 8-bit output tap)
           PACC_acq_ctrl.vhd                  (acquisition control block)
                    PNgen.vhd                            (MF/acquisition PN generator)
           Acq_intNdump.vhd                   (Acquisition Integrate & dump)
                    Acq_int_n_dump.edn                    (Integrate and dump adder)
           DBPSK_dec.vhd                      (DPBSK decoder)
           PACC_scrambler.vhd                          (De-scrambler)
           PACC_plcp.vhd                      (Header extraction/configuration)
                    CRC16_codec.vhd                    (CRC-16 decoding)
                    PNgen.vhd                            (demodulation PN generator)
       PACC_demodulation.vhd              (demodulation top entity)
                 demod_int_n_dump.edn         (adder integrate & dump)
                 DBPSK_dec.vhd               (DBPSK decoder)
                 demod_DQPSK.vhd                      (DQPSK decoder)
                 PACC_scrambler.vhd                   (De-scrambler)
       PACC_tx.vhd                           (Transmitter top entity)
                 PNgen.vhd                              (transmitter PN sequence generator)
                 PACC_plcp_tx.vhd            (Header generation block)
                    CRC16_codec.vhd                    (CRC-16 encoding)
                 PACC_scrambler.vhd                   (scrambler)
                 DBPSK_enc.vhd              (DBPSK encoding)
                 PACC_txdata_ctrl.vhd       (Transmission control block)
                    PNgen.vhd                            (PN sequence generator)
                    DBPSK_enc.vhd           (DBPSK encoding)
                    mod_DQPSK.vhd           (DQPSK encoding)
                    PACC_scrambler.vhd               (scrambler)
```

2.3.6 Testing

The modem testing has been performed in the test bed platform board. This board contains all necessary elements in order to fully test the modem, and the flexibility to test different components and interfaces for future expansions. The board contains a processor, which allows further testing of networking capabilities for future developments without hardware changes. This board was used to enable Rockwell Scientific the development of the modem while the hardware platforms was still undefined. By following this path Rockwell Scientific was allowed to develop a working modem that once the platform is defined, Rockwell Scientific can modify the modem to be optimized for such platform.

2.3.6.1 Testbed Description

The testbed board was developed to be flexible in the development and testing of a modem and its interaction with a network processor via a memory mapped interface. The board does not conform to any hardware size specifications, rather is a development platform for the concurrent modem development and final hardware definition. The testbed board is composed of a Xilinx XC1600E FPGA, an Intel SA-1110 processor, FLASH, SRAM, EPROMS, and clock generation circuitry for both processor and FPGA. The FPGA has multiple I/O for general probing purposes as well as future expansions. These I/O signals allow probing of different

internal signals for functionality and timing checks. The Testbed also contains interfaces to pattern generators and analyzers, which allow monitoring of several signals concurrently.

2.3.6.2 Test procedures

In order to verify the modem functionality we need to compare the simulation results with hardware test results. An external self-test block to the modem was generated. Setting a self test jumper and pressing the reset button in the board enables this block. The self-test first places the receiver modem in receive state, and after 120μs the transmitter will be enabled. The transmitter data is a counter, which increases its value for each transmitter data Byte. This self-test is enabled by placing a jumper in the board. When the jumper is removed, regular operation using the processor as the master device resumes. For the self-test procedure the configuration parameters are set via jumpers. These configuration parameters correspond to the type of encoding/decoding used (DBPSK/DQPSK), interleaver state, scrambler state and processing gain. The size of the payload, length field, is pre-determined to the largest packet size of 256 Bytes. Self-test operation is verified on a single board as well as two boards. In the single board setup the transmitter and receiver modems reside on the same FPGA. The transmitted I and Q channels are brought out to 2 pins which are connected externally via two cables to the input I and Q of the receiver side. The same test was performed for two boards. In this case the transmitter resides in one board, which now connects via the same cables for I and Q channels to the second board where the receiver resides. The two-board test case reflects a more realistic scenario, where the only connection between the transmitter and receiver is the channel. The channel in this case is represented by two wires for the I and Q signals. The processor interface has been implemented in the FPGA. The software side of this interface residing in the processor has not been tested. Rockwell Scientific has the capability to generate such an interface as well as the networking needed in order to make this interface as part of the necessary network. The software involved to control was not completed because of a reduction in funding.
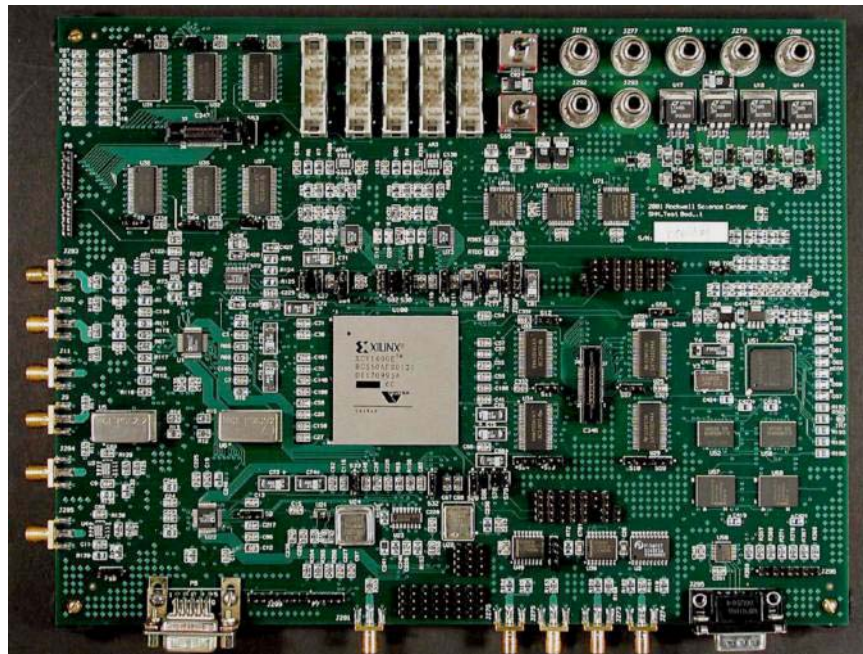
Figure 35: Photograph of the Testbed Hardware.

2.3.6.3 Implementation results

A major goal in the PAC/C effort is the power awareness. For this reason power consumption is an important parameter to be measured. Figure 36 shows all utilized resources in the FPGA. By extrapolating basic power consumption measurements, accompanied with data obtained from Xilinx data sheets some power reduction estimates can be done.

Target Device:	XV1600E	
Target Package:	BG560	
Target Speed:	-6	
Mapper Version:	virtexE	
Design Summary		

Number of errors:	0	
Number of warnings:	70	
Number of Slices:	1,161 out of 15,552	7%
Number of Slices containing		
Unrelated logic:	0 out of 1,161	0%
Total Number Slice Registers:	1,503 out of 31,104	4%
Number used as Flip Flops:	1,498	
Number used as Latches:	5	
Total Number 4 input LUTs:	1,712 out of 31,104	5%
Number used as LUTs:	1,636	
Number used as a route-thru:	76	
Number of bonded IOBs:	131 out of 404	32%
IOB Flip Flops:	6	
Number of Block RAMs:	12 out of 144	8%
Number of GCLKs:	3 out of 4	75%
Number of DLLs:	3 out of 8	37%
Total equivalent gate count for design:	245,301	
Additional JTAG gate count for IOBs:	6,288	

Figure 36: Implementation Resource Utilization Results.

Examining the implementation data we can clearly see the size of the FPGA is not appropriate for this design. Only a small percentage of the FPGA resources have been utilized. Given the FPGA technology the unused resources increase significantly the energy consumption. FPGAs have a quiescent power consumption, which is the power consumed just to keep the circuits programmed. This value increases with the size of the FPGA. For example a 1.6 Million-gate device is rated at 180mW quiescent power consumption, where a 200,000-gate device has estimated 29mW quiescent power consumption, 84% reduction. Our design after configuration of the device consumes 73mA at 1.8V, 131mW quiescent power. The modem design would fit in a 200,000-gate device, which taking the rated power values would reduce the quiescent power consumption by 84% to 12mA. In the future, once the implementation hardware platform has been defined, accurate power measurements should be performed. At this point such measurements are out of the scope of the project, and difficult to measure in the testbed.

3   FINAL STATUS REPORT ON
MIDDLEWARE, TOOLS, AND TECHNIQUES

3.1   Introduction

The purpose of this task was to examine techniques for energy management that leverage the interactions between hardware and software on a single node as well as network-wide interactions among many collaborating nodes in a sensor field. This research was performed through a combination of simulation-based analysis and platform studies using commercial and research platforms. For the single node, the research focused on the development of a power aware Real-Time Operating System (RTOS) and other power management middleware APIs. Techniques for node-level power management are discussed in Section 3.2. At the field-level, the areas of investigation include communications techniques such as dynamic modulation scaling and protocols for collaborative field behavior such as coverage topology management. Techniques for network-wide power management are discussed in Section 3.3. Finally, the concept studies in node-level and field-level power management utilized and extended a suite of sensor network simulation tools developed at UCLA. The SensorSim tool environment is described in Section 3.4.

3.2   Techniques for Node-Level Power Management

The current state of the art in software-transparent hardware and firmware power management is limited to shutting down the CPU and peripheral components after a certain idle time. This technique fails to exploit application knowledge of timing constraints, usage history, and traffic patterns. It also ignores the possible existence of "power-speed control knobs" (e.g. CPUs with dynamically controlled frequency and voltage). Therefore, the current methods cannot fully utilize the dynamic range of power management that could be provided by the lower layers. This task examined new approaches to address these issues. Since knowledge about timing, usage, and traffic is often available only at run-time, the logical place for making power management decisions is in the Real-Time OS (RTOS) resource scheduler. Key to our approach is task management that guarantees "JIT" (just in time) power through coordinated scheduling and power management of resources by the RTOS at a microsensor node.

3.2.1   Power Aware Resource Scheduling

Wireless systems have various resources that consume power including CPUs, radios, and sensors. However, traditional RTOS schedulers have focused on the CPU resource from the perspective of time (i.e. statically or dynamically scheduling multiple tasks so that timing is met). Some schedulers manage other resources such as disks, but none manage resources like radios and sensors, which are critical to the extended operation of a sensor node. In any case, power is generally ignored. Key to a power-aware sensor node is power management of all node resources. This effort expanded time-based static and dynamic scheduling to include adaptive power-management of all accessible sensor node subsystems. It examined both static scheduling and variants of dynamic scheduling, including fixed priority preemptive scheduling (e.g. rate monotonic) and dynamic priority preemptive scheduling (e.g. earliest deadline-first). The basic approach exploits slack time in the schedule to shutdown a resource or to operate it at a lower-power lower-speed setting. Slack times are gathered by static analysis of the task set and by

timing constraint specifications (rates and deadlines), as well as runtime statistical analysis of resource usage patterns gathered by special hardware or software "monitors". We investigated the problem of prediction of resource usage activity, which is essentially a timing series prediction problem, and explored approaches such as Kalman filtering, voting by multiple experts, and training a parametric model (e.g. a hidden-Markov model). A meaningful strategy for scheduling resource mode changes also requires knowledge of the power and/or time cost associated with the change. For example, changing the radio from idle to transmit mode would require some latency and would consume energy (e.g. for the frequency synthesizer to stabilize, or for GPS systems to acquire a lock). Therefore, the RTOS must decide whether the extra power and latency cost is worth the change. Before shutting down a radio, the RTOS must determine if wake-up latency can be tolerated (or, if it could be hidden by pre-wakeup). The prediction accuracy for scheduling resource mode changes can be further enhanced by making use of information that can be provided by the application. For example, a radio may not know when the next packet will be received or transmitted, and it may be hard for the radio monitor to learn this. However, applications potentially have this information and can provide it to the RTOS when admitted for scheduling, or as they run. Figure 37 shows the hardware resources and software layers for coordinated node-level power management.
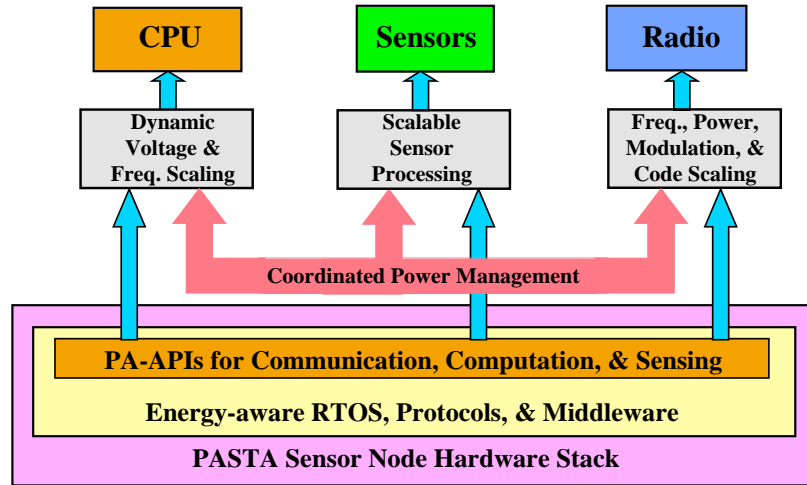
Figure 37: Coordinated Node-Level Power Management

3.2.2 *Power-aware API for RTOS-driven CPU Power Management*

While working on adaptive power-fidelity schemes the research team had identified the need for a common software framework to: a) allow an apples-to-apples comparison of shutdown and dynamic voltage scaling (DVS) power management schemes, b) provide a standard interface for developers of power-aware applications, and c) allow easy porting of implementations to multiple embedded platforms. The team developed a power-aware API that provides standard software interfaces to exchange power and performance information among power-aware hardware, operating system, and applications to support task scheduling, admission control, and run-time adaptation. Table 18 provides a list of power aware API functions.

Table 18: Power Aware API Functions

paapi_sched_create_task_type (real time info)	Create a task type given the real time constraints.
paapi_sched_create_task_instance (type)	Create a task instance given the type.
paapi_sched_app_started (task instance)	Tell the operating system that the task started a new

	execution.
paapi_sched_app_done (task instance)	Tell the operating system that the task is done with the execution.
paapi_os_timetick_get_minimum_resolution(nsec)	Find out what is the minimum resolution the OS tick can be set to met the timing constraints.
paapi_os_timetick_time_to_ticks(nsec)	Convert the time restriction, given in nanoseconds, into OS ticks.
paapi_sched_set_time_prediction(task instance)	Tell the operating system about timing remaining prediction (specific to predictive scheduling).
paapi_sched_get_time_prediction(task instance)	Get the time prediction given by the operating system (specific to predictive scheduling).
pahal_initialize_processor_power_management()	Initialize processor power management internal data structures.
pahal_processor_get_freqlevels_info()	Get processor frequency level information.
pahal_processor_get_currfreq_info()	Get current frequency level info.
pahal_processor_pre_set_frequency(current, new)	Function that is called before changing processor frequency (platform dependent).
pahal_processor_set_frequency(current, new)	Function to actually change processor frequency.
pahal_processor_set_frequency_and_voltage(new_frequency)	Function to actually change frequency and voltage.
pahal_processor_post_set_frequency(current, new)	Function that is called after changing processor frequency (platform dependent).
pahal_processor_get_states_info()	Get information about low power states of the processor.
pahal_processor_set_state()	Set the processor to a specific low power state.
pahal_processor_cycles_to_nsec()	Convert cycles to nanoseconds.
pahal_processor_nsec_to_cycles()	Covert nanosecond to cycle.
pahal_battery_get_info()	Get battery related information (for battery based devices).
pahal_battery_get_percentage_available()	Get percentage of battery available in the system.

The power aware RTOS implementation includes three different schedulers. The first is a rate-monotonic scheduler (based on static priority based preemptive scheduling) with shutdown. The second is a rate-monotonic scheduler with shutdown and per-task static slowdown factors according to which voltage and frequency are set at task context switch time. The third is a rate-monotonic scheduler with shutdown, per-task static slowdown, and a dynamic slowdown based on run-time task execution time prediction. The latter two schemes are based on algorithms developed at UCLA under this project. The API was implemented on top of eCos, an open-source RTOS, as shown in Figure 38. Two different platforms are supported: the StrongARM-based iPAQ and the Intel XScale 80200 Evaluation Kit. The iPAQ is capable of shutdown and frequency scaling. The evaluation kit allows software-controlled voltage scaling when combined with a Maxim 1855 Evaluation kit (MAX1855EVKIT) as power supply (shown in Figure 39). The RTOS software with power-aware API and hardware reference design specifications for the XScale dynamic voltage scaling testbed are available to the research community via the web: http://nesl.ee.ucla.edu/projects/pads.

Figure 38: Power Aware API Layers	Figure 39: Dynamic Voltage Scaling Testbed
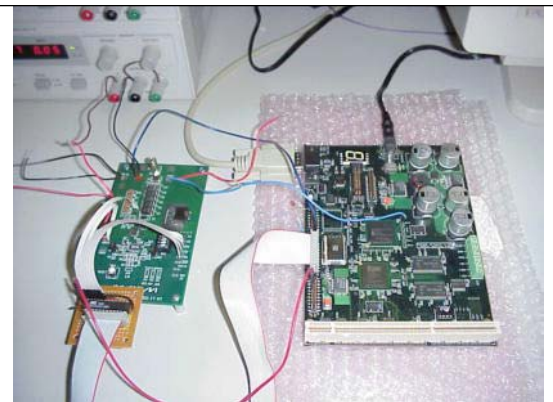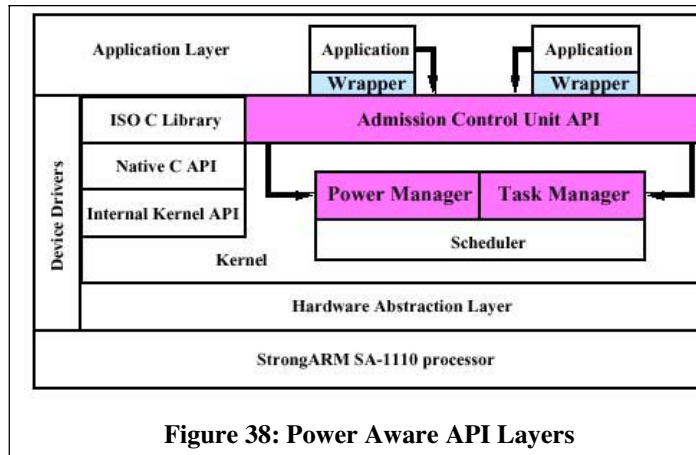
3.2.2.1 Power Analysis of XScale with Dynamic Voltage Scaling

Using the dynamic voltage scaling testbed described above and the power aware software framework, UCLA performed a detailed power analysis of XScale processor at the instruction level. The processor has a dynamic power range of approximately 5X as frequency scaled from 333 MHz to 733 MHz, with the voltage being changed concurrently. However, in some cases the device could be significantly over-clocked for a particular voltage setting. Figure 40 shows power when the processor is idle (NOP). Figure 41 and Figure 42 give power ranges for integer addition and floating point multiply. Figure 43, Figure 44, Figure 45, and Figure 46 provide memory read, memory write, peripheral read, and peripheral write, respectively.
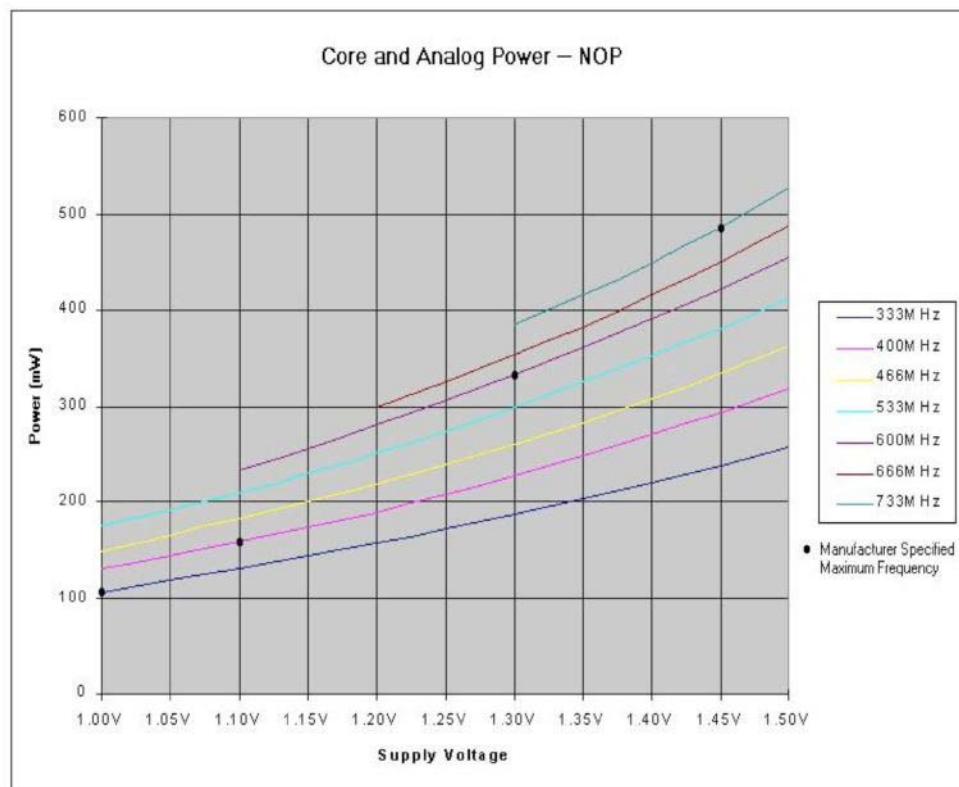
Figure 40: XScale NOP Power
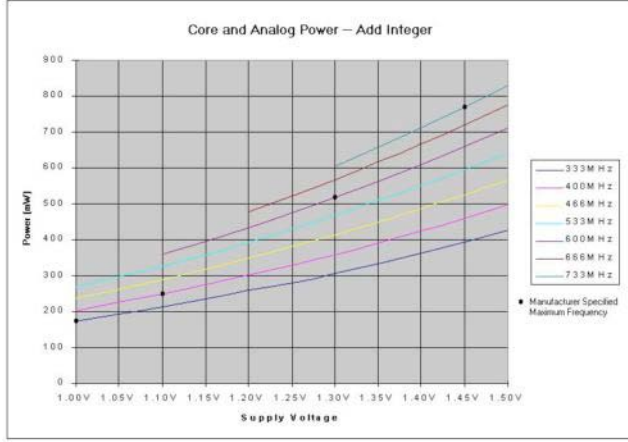
Figure 41: XScale Add Integer Power
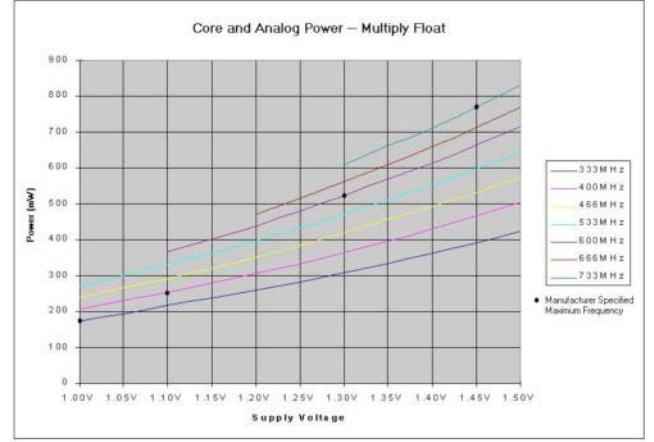
Figure 42: XScale Multiply Float Power
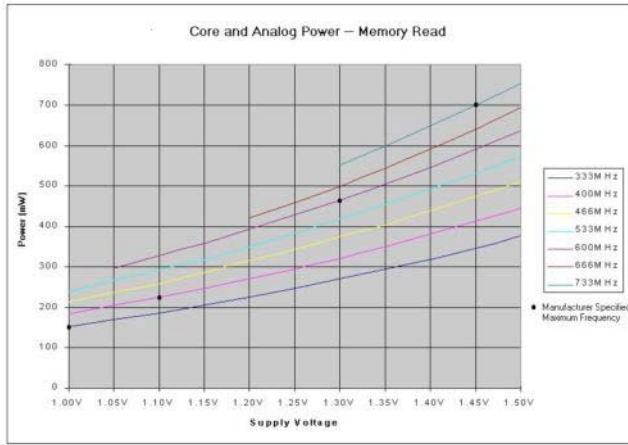
Figure 43: XScale Memory Read Power
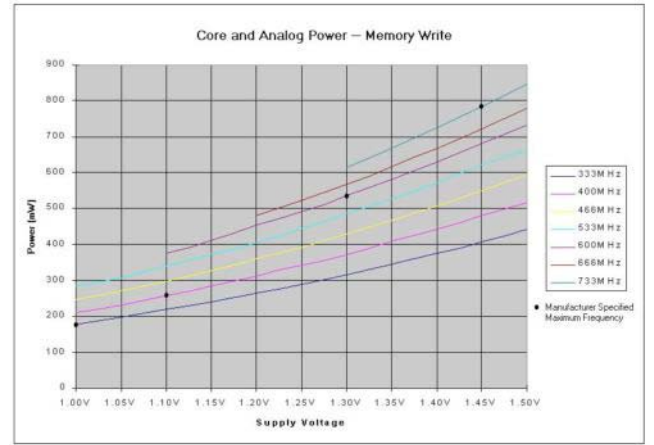
Figure 44: XScale Memory Write Power
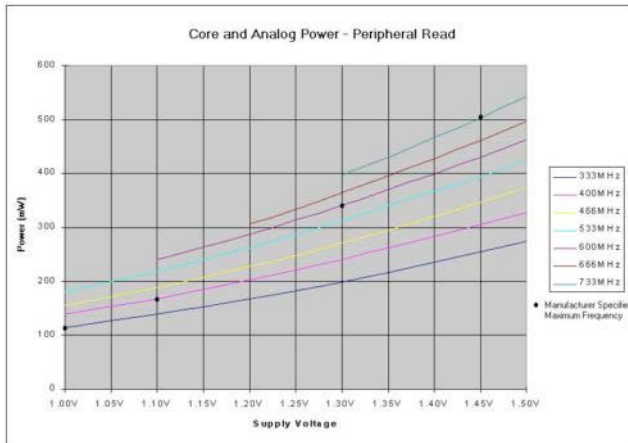
Figure 45: XScale Peripheral Read Power
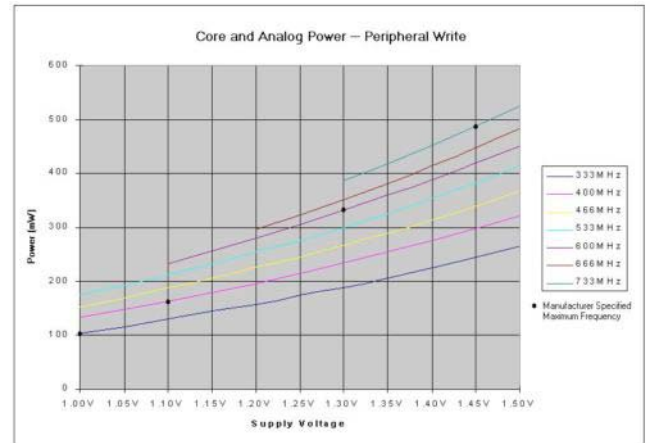
Figure 46: XScale Peripheral Write Power

The measurement approach used by the UCLA team was to insert numbered *power measurement regions* (PMR) in the source code using macros (PMR_START, PMR_END) from the power aware API. The power aware runtime system logs the PMR and triggers the data acquisition card (DAQ) with a GPIO. A PC-based analysis tool matches the PMR and DAQ logs, calculates averages and total energy in each PMR. These basic power/performance datasets

were used to generate thresholds for predictive task scheduler research and the software infrastructure was used to validate the scheduler algorithms.

3.2.3 *Predictive Dynamic Voltage Scaling for Adaptive CPU Fidelity*

Analysis of execution times for a variety of signal processing tasks in embedded systems shows that the *worst-case execution time* (WCET) is often several times longer than *best-case execution time* (BCET). For example, Figure 47 shows the difference in MPEG frame decoding time and Figure 48 plots varying times for audio speech decoding. However, wireless systems such as sensor networks need to be resilient to packet loss. *A crucial observation is that missed deadlines in many computation tasks on a sensor node are no different than noise in the radio or sensor: it would result in a radio or sensor packet being dropped.* This allows a scheduler to aggressively predict task instance runtime and scale the dynamic voltage accordingly.
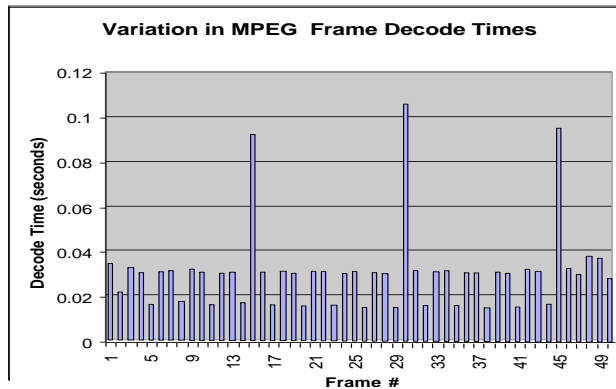
Figure 47: MPEG Decode Times
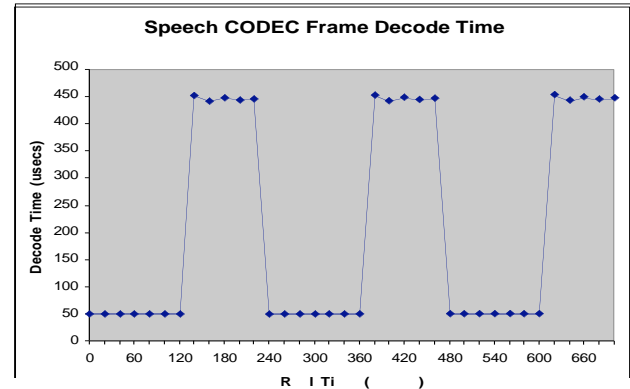
Figure 48: Speech CODEC Decode Times

The result of proactive dynamic voltage scaling with prediction on a variety of signal processing tasks was a 75% energy reduction over worst-case non-predictive voltage scaling with a negligible loss in fidelity (up to 4% deadline misses). Figure 49 compares energy consumed by shutting down hardware on task completion, a WCET non-predictive dynamic voltage scaling scheduler, and the predictive dynamic voltage scaling scheduler based on BCET. These results demonstrate that many applications have "energy-speed" or "energy-accuracy" control knobs that allow a given task to operate at lower energy but over a longer time or with lower fidelity. An important feature of the power aware API research was exposing these knobs to the application developer so they can be dynamically managed at runtime.
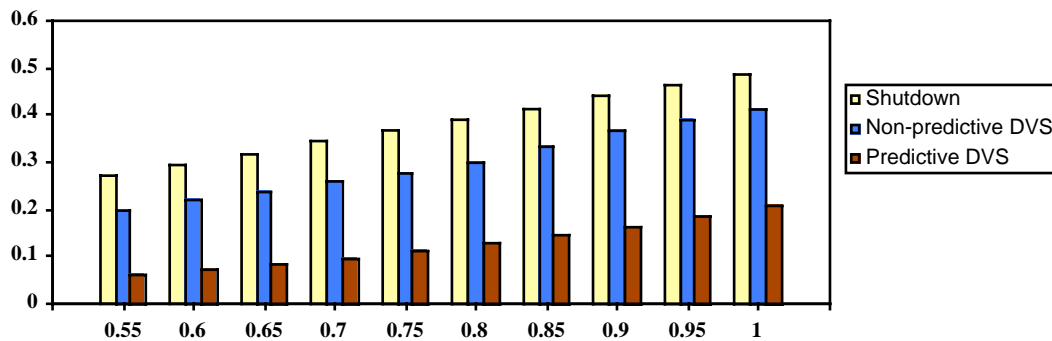
Figure 49: Predictive Dynamic Voltage Scaling Energy

3.2.4 Battery Lifetime Management

An interesting result of this research was the impact of task scheduling on achieved battery capacity. An ideal battery's capacity doesn't vary with the discharge rate. However, real batteries have discharge rate dependent characteristics such as the reactants diffusion process, the rate of electrode reaction, and the battery's internal impedance. The maximum battery capacity is only achieved at a low discharge rate.

Operating Mode	Total Lifetime	Capacity (in mAh) consumed from Battery	Capacity (in mAh) delivered to Sensor	Energy (in Joules) consumed from Battery	Energy (in Joules) delivered to Sensor	DC/DC converter efficiency
Tx	.33 hr	8.1	4.1	70.0	49.1	
Rx	.88 hr	15.8	7.9	135.2	96.1	71%
Idle	1.2 hr	18.0	9.0	154.0	109.0	71%
Sleep	7.0 hrs	59.1	28.8	505.1	350.7	69%

Figure 50: Achieved Battery Capacity Versus Operating Mode

The original Mote microsensor developed at UC Berkeley uses an unregulated voltage supply directly from the battery. Unlike operating with a voltage regulator, this exposes the battery to dynamic variations in current discharge based on operating mode of the microsensor. For example, in Figure 50 the total energy extracted from the battery varied from 70 Joules continuously transmitting to 505 Joules in sleep mode. A number of battery-aware scheduling approaches were investigated to find optimal performance to maximize total service during the lifetime of the battery. The best approach achieved an 80% improvement in total number of bits transmitted over non-battery-aware techniques. Figure 51 shows the measured battery capacity versus discharge power (left), data rate in kilobits per second versus average power (center), and total number of bits versus average power (right). This indicates that there is an optimal data rate of about 5kbps to maximize Mote battery lifetime.
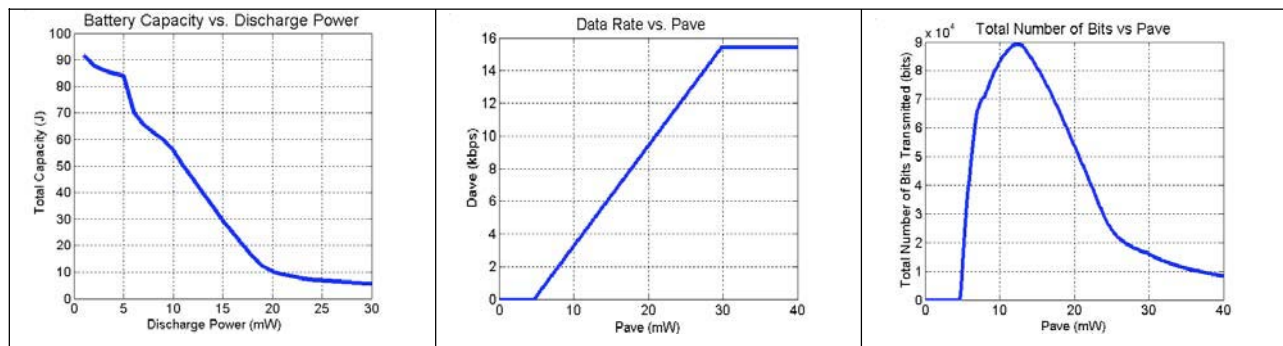
Figure 51: Data Rate and Battery Capacity Comparisons

3.3   Techniques for Network-Wide Power Management

This task also investigated techniques for network-wide distributed power management that complemented the power aware RTOS and middleware described above. Distributed power management is essential to sensor fields because of the need to collaborate and communicate for most sensing applications. There were two main areas of investigation: 1) communication techniques designed to save energy on the wireless links and multi-hop routes, and 2) collaborative distributed resource allocation strategies to maximize effective sensor field lifetime while maintaining continuous coverage.

3.3.1 Energy-aware Packet Scheduling with Dynamic Modulation Scaling

The notion of Dynamic Modulation Scaling (DMS) was based on the observation there is a strong analogy between voltage scaling of a CPU and bit-per-symbol/bit-per-second scaling for a radio. Energy and delay of data transmission depends on modulation settings. This was studied in detail for Quadrature Amplitude Modulation (QAM) radios with adaptive bits-per-symbol. The tradeoffs for QAM are to adapt b (number of bits per symbol) and operate at maximum R_s that can be implemented efficiently. Similar tradeoffs are possible for other scalable modulation schemes Phase Shift Keying (PSK), Differential Phase Shift Keying (DPSK), Amplitude Shift Keying (ASK), and Orthogonal Frequency Division Multiplexing (OFDM).
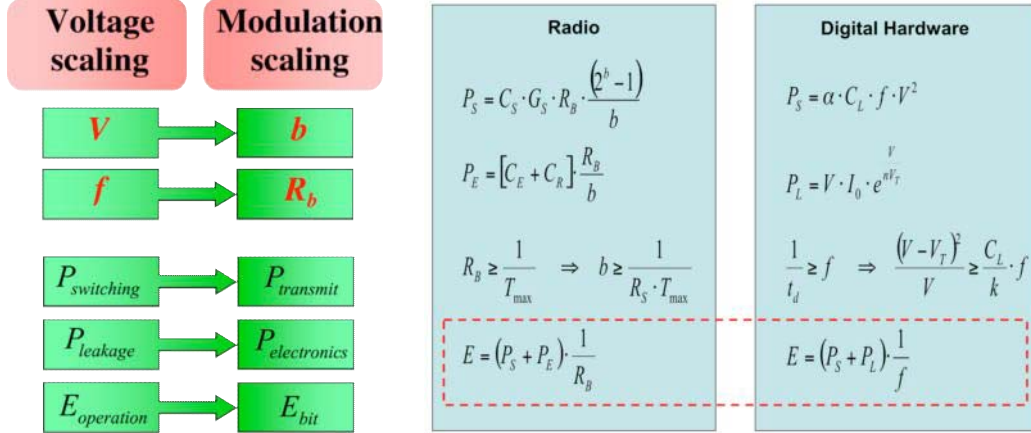
Figure 52: Voltage Scaling Versus Modulation Scaling

3.3.1.1 Queue-Based Dynamic Modulation Scaling

Moreover, packet scheduling in radios is analogous to task scheduling on processors. A key difference is that packets, unlike tasks, are not preemptible. A number of packet scheduling schemes were investigated for wireless links using dynamic modulation scaling. The first to be investigated was Queue-Based Dynamic Modulation Scaling. In this technique, modulation settings were set according to the packet queue status including number of packets and total size of transmit buffer on the sending end. This is the DMS counterpart to simple queue-based schemes commonly used in task schedulers.
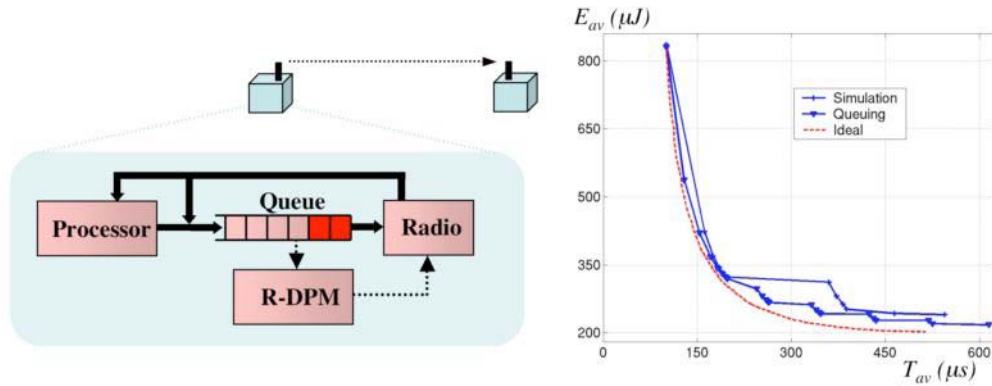
Figure 53: Queue-Based Dynamic Modulation Scaling

3.3.1.2 Energy-Aware Real-Time Packet Scheduling

The second scheme, Energy-Aware Real-Time Packet Scheduling, exploits variation in packet length to perform aggressive dynamic modulation scaling in real-time. This technique handles the case of multiple applications sending streams of periodic but varying length packets with deadline constraints. It is suitable for sensor nodes talking to a local gateway. Analysis of this approach demonstrated up to a 60% reduction in transmission energy. Similar packet scheduler approaches were investigated, such as an energy-aware variant of the commonly used weighted fair-queuing based packet schedulers. In the weighted fair-queuing model, weights are associated with the various packet flows and the scheduler allocates the available bandwidth to those flows with non-empty queues in proportion to their weights over arbitrarily small time intervals. Conventionally the scheduler decides which flow to transmit a packet from, and when. The new scheduler, energy-efficient wireless fair queuing (E2WFQ), exploits the dynamic modulation-scaling knob to decide at what speed a packet should be transmitted. It does so by continually monitoring the status of the packet queues, and calculates how fast a packet needs to be transmitted based on the packets ahead of it and the deadline. The reduction in energy depends on the traffic condition parameters such as link utilization and burstiness. There is little gain from the scheme if the link utilization is near 1 (i.e. there is little head room left to exploit DMS as the channel is always busy even with the radio at maximum speed) or if the burstiness is high. However, gains of 3-4X are observed under typical traffic conditions and radio speeds.
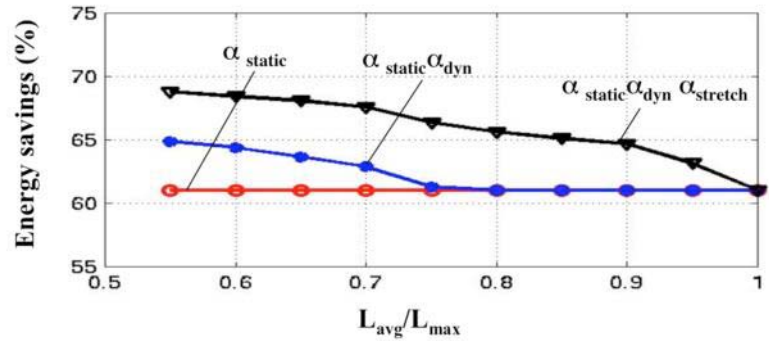
3.3.1.3 Dynamic Code Scaling

In addition to Dynamic Modulation Scaling, the research team also investigated Dynamic Code Scaling (DCS) by changing the rate of the bit-level forward error correction (FEC) within the same DMS framework. Both DMS and DCS offer a convex energy-throughput control knob that energy-aware packet scheduling can exploit. Figure 54 shows the compares the energy per useful bit for fixed and dynamic code rates.
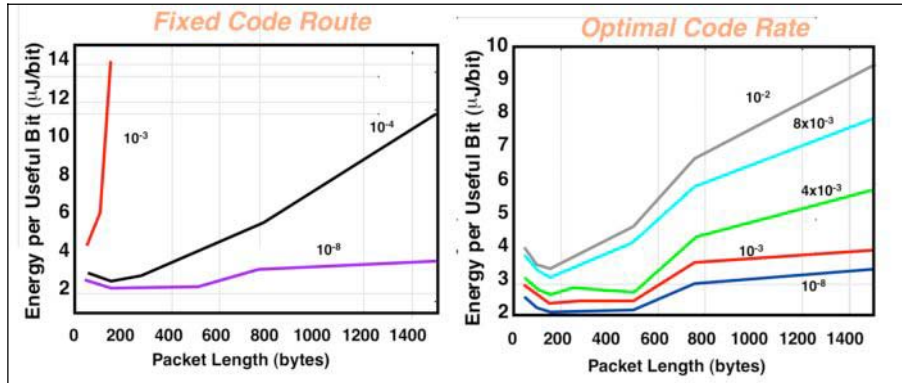
Figure 54: Energy per Useful Bit with Dynamic Code Scaling

3.3.2 Sparse Topology and Energy Management

Sparse Topology and Energy Management (STEM) improves network lifetime by exploiting the fact that, most of the time, the network is only sensing its environment waiting for an event to happen. The motivation is that the energy consumption in sensor network is typically dominated by the node's communication subsystem. It can only be reduced significantly by transitioning the embedded radio to a sleep state, at which point the node essentially retracts from the network topology. Existing topology management schemes have focused on cleverly selecting which nodes can turn off their radio, without sacrificing the capacity of the network. For this they exploit extra node density beyond what is needed for communication connectivity. However, in sensor networks preserving communication capacity at all times is not a good idea. Alleviating the restriction of network capacity preservation yields extensive energy savings at the expense of an increased latency to set up multi-hop paths. The basic STEM scheme is based on using two radios at each node: one is a regular data radio while the other is a wake-up radio. The goal of having two radios is that the wakeup radio could be much simpler with a much lower data rate and lower power on the receive side. However, the scheme works even with two identical radios with no special differentiation in capabilities. It can also work with a single radio with two channels or a single radio with one channel with time synchronization at some degradation in performance. In the two-radio scenario, one radio operates in the data plane, and the other in the wakeup plane. As shown in Figure 55, receivers in the wakeup plane periodically listen for beacon packets on a very low duty cycle (0.1%). An initiator issues a train of beacon packets until the target acknowledges. The data packet then commences on the data plane.
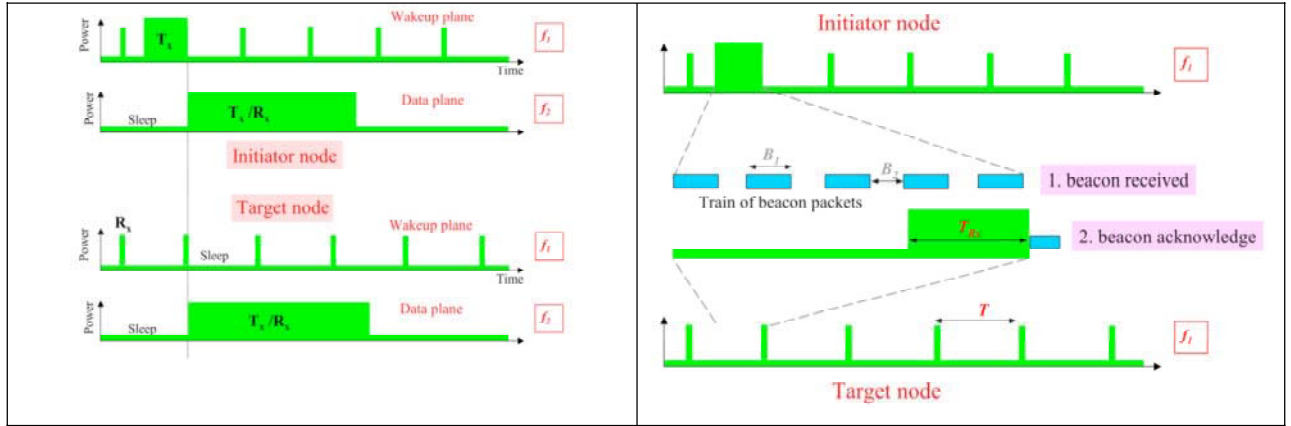
Figure 55: Operation of STEM

As shown in Figure 56, the energy savings for STEM is dependent on the fraction of time the communications subsystem spends in the forwarding state. In most monitoring sensor systems, data packets are highly infrequent, so the monitoring state dominates.

- Setup latency $\overline{T}_S \approx \dfrac{T + \frac{2}{3} T_{Rx}}{2}$ *Appropriate choice of interval sizes*

- Energy savings $\dfrac{E}{E_0} \approx \dfrac{T_{Rx}}{T}$ *Mostly monitoring state: $\alpha << 1$*
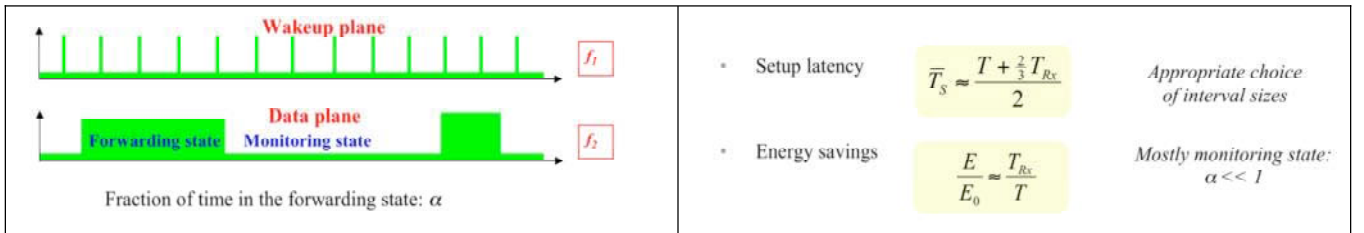
Figure 56: STEM Performance Analysis

Figure 57 shows STEM performance results from simulation analysis. The approach has been validated on a testbed consisting of iPAQs equipped with 802.11b radios as the primary data radio and a mote microsensor with the RFM TR1000 radio acting as the wakeup radio. The motes were connected to the iPAQs over the serial ports. The STEM algorithm performed as predicted by the simulation models.
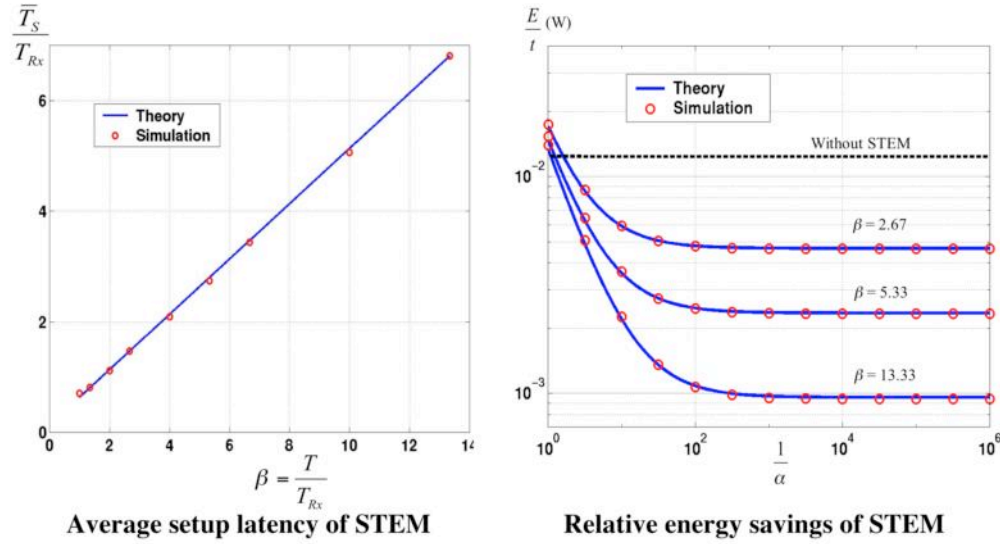
Figure 57: STEM Simulation Analysis

3.4   Sensor Networking Simulation and Planning Tools

Under the DARPA SensIT program, ISI and UCLA were funded to develop a simulator for networking protocol and systems analysis of distributed sensor networks called *SensorSim*. An *ns*-based simulation platform was developed that modeled media access control (MAC), link, and routing protocols used in the SensIT program. A unique and useful feature of this simulation tools was the capability to perform hybrid simulation with both real and simulated nodes in the same network. In this way, a much larger virtual sensor network could exercise an actual instrumented hardware node with appropriate traffic. The PADS effort extended this *SensorSim* simulation infrastructure with several new capabilities, including:

1) A capability to explicitly model the hardware hierarchy of a networked sensor node, and its power behavior. The original *ns* simulation tool allows modeling of protocol agents and application agents at each networked node, but there was no provision of modeling the power behavior of hardware resources such as the processor, sensors, radio, battery etc. The radio, processor, sensors, and other power consumers are now modeled as hardware resources that have multiple states of operation (sleep, idle, active etc.), each with different levels of performance and power consumption, and time and power costs associated with transitions from one state to another. Each resource has an associated API so that the higher level systems agents (protocols, OS, applications) modeled in *ns* can 1) use the resource for some interval of time, 2) cause the state of the resource to change, and 3) change their behavior according to system parameters impacted by the state. On the producer side, the battery model includes the dependence of battery lifetime on discharge current rate and discharge current profile (constant discharge vs. pulsed discharge). Datasets for these models were generated from the power characterization from instrumented platforms such as the Berkeley Mote, iPAQ, and XScale evaluation board.

46

2) Modeling of protocol and RTOS level power management. The simulation models of higher-level agents (e.g. protocols, applications) in this framework can identify the hardware resources they need, the usage interval, and the state of operation. Thus, the amount of CPU time and energy used on behalf of a routing protocol is accounted for. Agents incorporate dynamic power management policies for the various resources, and thereby control the transition between states. For the radio, power management is a responsibility of the MAC protocol, with cooperation by the RTOS.

3) Power-aware hybrid simulation/emulation. The simulation framework provides a unique capability whereby part of the network would consist of simulated nodes, while the rest of the network would consist of real nodes. Specifically, the framework allows a simulated network to be connected to a real network of sensor nodes via a gateway so that only the part of the network whose power and performance needs to be studied in a controlled fashion is simulated while the real nodes could efficiently abstract the rest.

A diagram of the SensorSim architecture is shown in Figure 58. The models developed under this effort have been incorporated into the *ns* network simulator. It is also available at the UCLA PADS project web site: http://nesl.ee.ucla.edu/projects/pads.
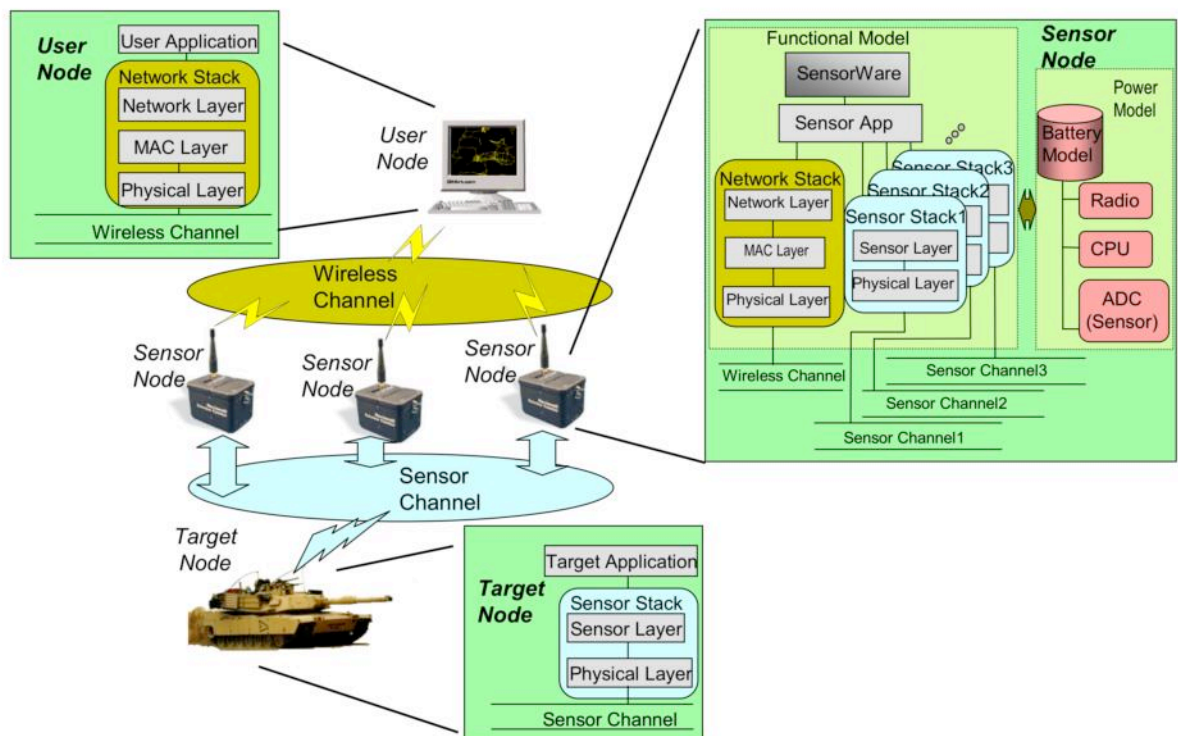
Figure 58: SensorSim Architecture

4   FINAL STATUS REPORT ON ALGORITHMS

4.1   Introduction

Consistent notion of identifying accuracy/energy "knobs" described in the previous sections on hardware and software, power aware algorithm development uses the same approach. In the PASTA effort, the algorithm team realized that there are three distinct phases in power aware algorithm development: analysis, optimization, and introspection. In the analysis (or "knobs") phase, the goal is to identify algorithm input parameters that can be adjusted and determine their impact on accuracy and energy. In this phase, the input parameters are exposed to manipulation by the software. In the second phase, other optimizations unrelated to input parameters are investigated such as eliminating domain transforms or introducing domain transforms to simplify the computation. The third phase, called introspection, incorporates a goals-focused "knob" into the algorithm such as accuracy, false alarm-rate, or target power consumption. The "knob" can be set by the application designer at design time, or adjusted dynamically in real-time by some rules-based agent monitoring environment activities and remaining hardware resources (battery lifetime, for example).

Figure 59: Three Stages of Power Aware Algorithm Development

4.2   Acoustic Beamforming

ISI obtained an acoustic beamforming algorithm from the Army Research Laboratory for power aware analysis. In addition to the algorithm in MATLAB (and ported to C by MIT), the application baseline package contained one and two vehicle datasets along with ground truth. The acoustic array contained a total of seven microphones; six evenly spaced on an 8-foot diameter circle and the seventh in the center. The acoustic array is shown in Figure 60. With permission of ARL, ISI distributed this baseline algorithm throughout the PAC/C community.

Figure 60. ARL Baseline Acoustic Array

4.2.1 Acoustic Beamforming Knobs

Acoustic beamforming algorithms estimate the line of bearing (LOB) to a distant acoustic emitter by shifting signals from microphones at known relative locations to form beams from selected directions. The LOB is a byproduct of the optimal reconstruction of the signal from an emitter by shifting and adding the signals from a number of microphones. The background noise and sounds from other emitters will be reduced in proportion to the number of microphones. The accuracy of acoustic beamforming depends on a number of parameters, including: 1) the number of microphones, 2) number of acoustic samples, and 3) number of beams or search angles. Each of these parameters has a unique impact on accuracy and energy of the system.
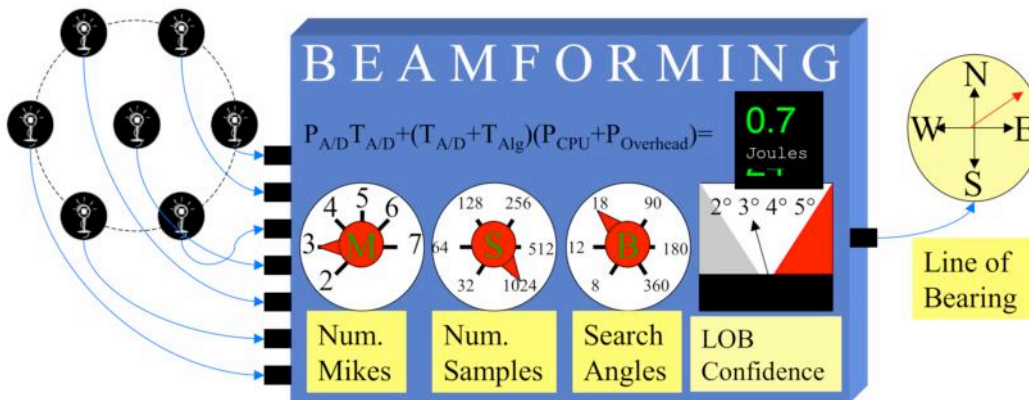
Figure 61: Beamforming "Knobs"

4.2.1.1 *Number of Microphones (M)*

The number and placement of microphones is principally a hardware design parameter but can also be used as an algorithmic parameter by selecting a subset of the signals collected. The system energy required for beamforming is *proportional* to the number of microphones, but there are negligible improvements in accuracy due to adding more microphones at the same

49

radius. The analysis is limited to nodes supporting at least two microphones. Although single-microphone nodes could collaborate to determine the direction to an emitter, this case was not considered due to the communication requirements to move raw data between nodes, of 10-us synchronization between nodes, and of 5-mm accuracy in relative positioning of microphones. Although the ambiguities resulting from two-microphone beamforming would require collaboration, only a trivial amount of data need be exchanged, synchronization to a fraction of a second is sufficient, and the required accuracy of relative positioning is similarly relaxed. As shown in Figure 62, the root mean square (RMS) error for three microphones is approximately 0.2 degrees less accurate than seven microphones.
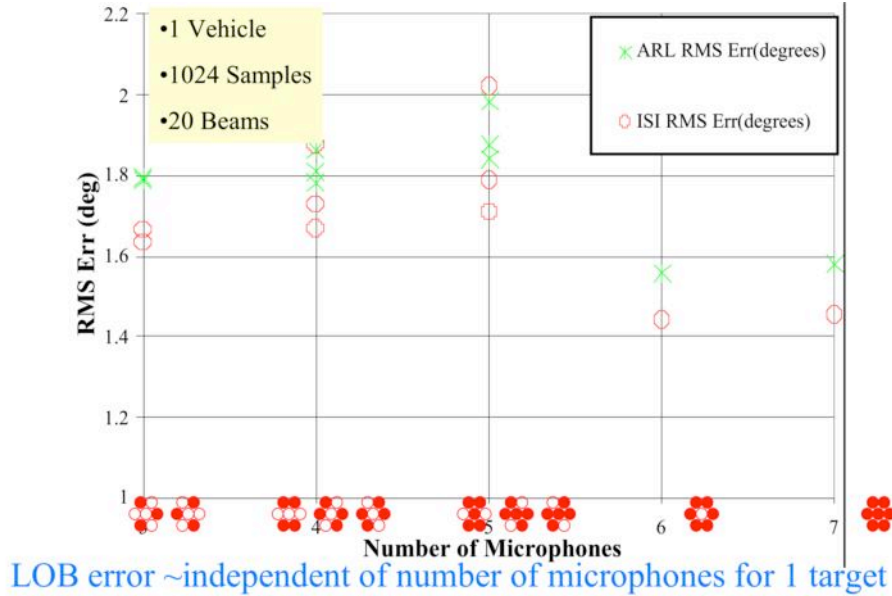
Figure 62: RMS Error Versus Number of Microphones

4.2.1.2 *Number of Acoustic Samples (S)*

The baseline HIDRA platform used in this project had a fixed sampling rate of 1024 Hz, or 1kHz. The system energy required for beamforming is *proportional* to the number of samples simultaneously collected from each microphone. The number of samples collected per second for each microphone is typically 1 kHz for acoustic tracking of vehicles to facilitate beamforming with the spectrum above 250 Hz attenuated to filter out wind noise. This was implemented as an analog anti-aliasing filter. Figure 63 shows the RMS error versus the number of samples processed. As expected, the accuracy significantly begins to degrade below 128 samples.
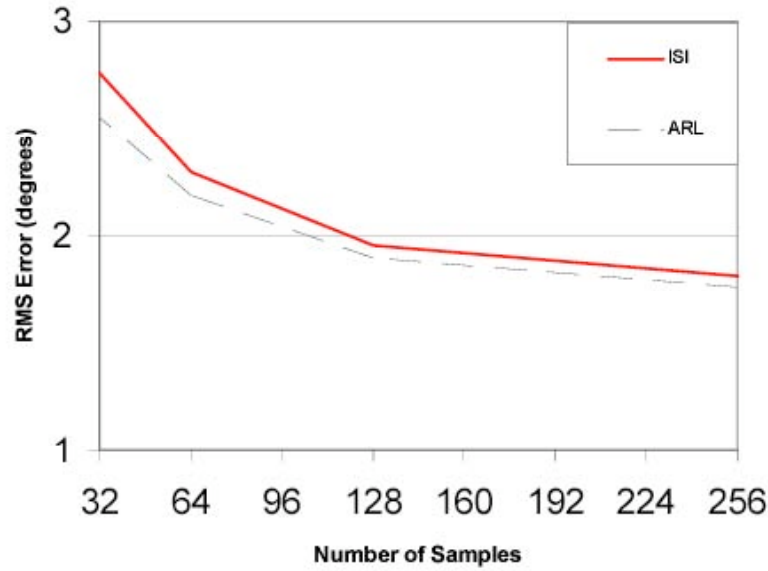
Figure 63: RMS Error Versus Number of Samples

4.2.1.3 Number of Beams (B)

The beam power is computed at a number of evenly spaced search angles, and interpolated by a parabolic fit to estimate the angle with maximum power. The number of beams only affects the execution speed of the algorithm, *not data acquisition*. The system energy for beamforming is linear with, not proportional to, the number of beams searched. As shown in Figure 64, the RMS error is nearly constant over 15 beams.
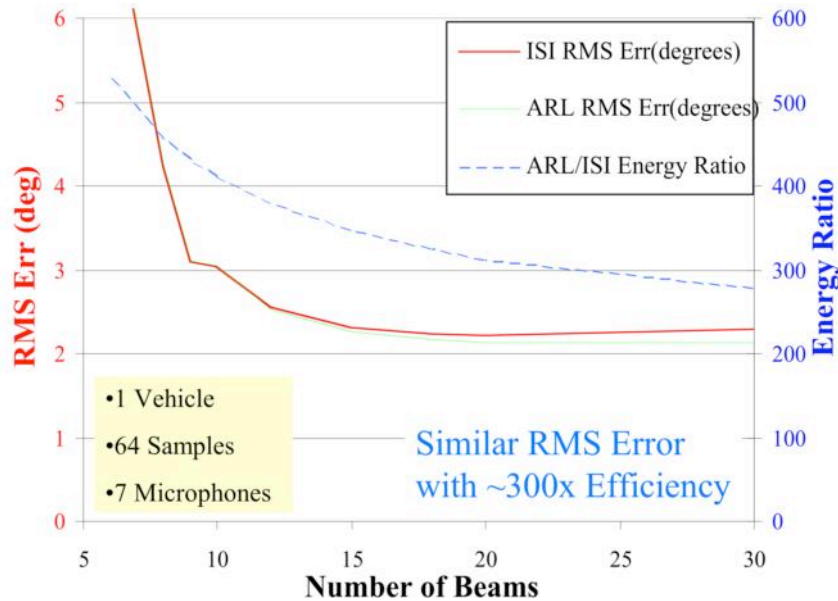
Figure 64: Accuracy and Energy Versus Number of Beams

4.2.2 Algorithm Optimizations

In addition to the power aware algorithm "knobs" described above (microphones, samples, beams), the team investigated the impact of converting from floating point to integer math and eliminating the data transform to the frequency domain. Since floating point is emulated in software on the XScale processor, converting to integer math had the most significant impact on power – approximately 20X. Eliminating the Fast Fourier Transform (FFT) to the frequency domain and performing a time-domain beamform (Figure 65) yielded another factor of 1.5X.
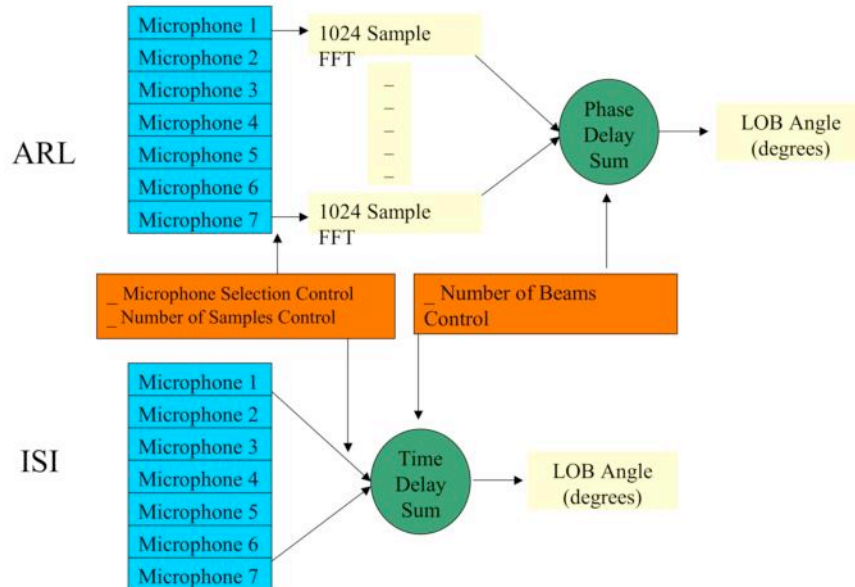
Figure 65: Frequency Domain Beamformer (top) and Time Domain Beamformer (bottom)

4.2.3 Acoustic Line Of Bearing Results

Figure 66 shows the acoustic line of bearing energy according to the RMS error. Using this "introspection knob", the desired accuracy or the desired energy can be chosen independently and the algorithm parameters (number of beams, number of samples, number of microphones) are adjusted to achieve the best result. It is clear in this graph, that the RMS errors less than three degrees consume a disproportionate amount of energy. This suggests that an adaptive multi-resolution search strategy would have a significant impact on total system power. The blue line (dotted) shows the ISI algorithm energy performance relative to the original ARL baseline implementation. The ISI version achieved between 180X and 850X energy improvement (depending on selected RMS error) on the same hardware.
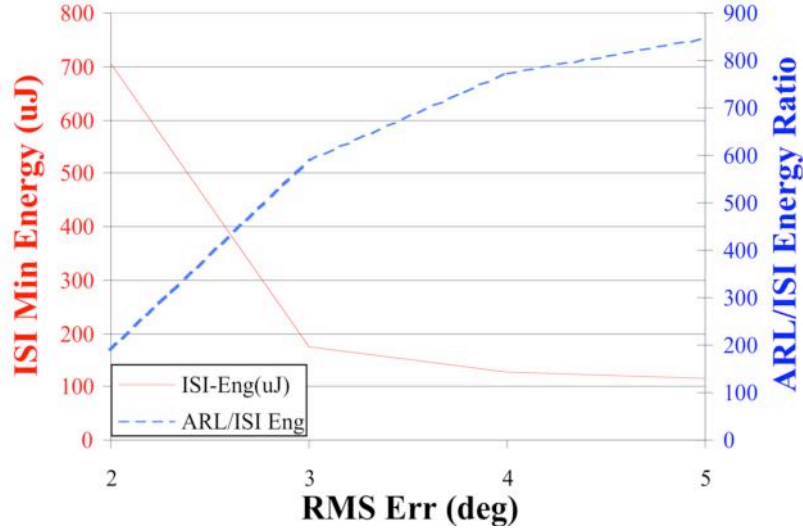
Figure 66: Acoustic Line of Bearing Energy Versus RMS Error

Figure 67 plots ISI and ARL location estimates against the GPS-derived ground truth supplied with the algorithm. The two versions of the algorithm can be tuned to be identical in performance. The primary difference is that the ISI version can relax accuracy constraints to conserve more energy.
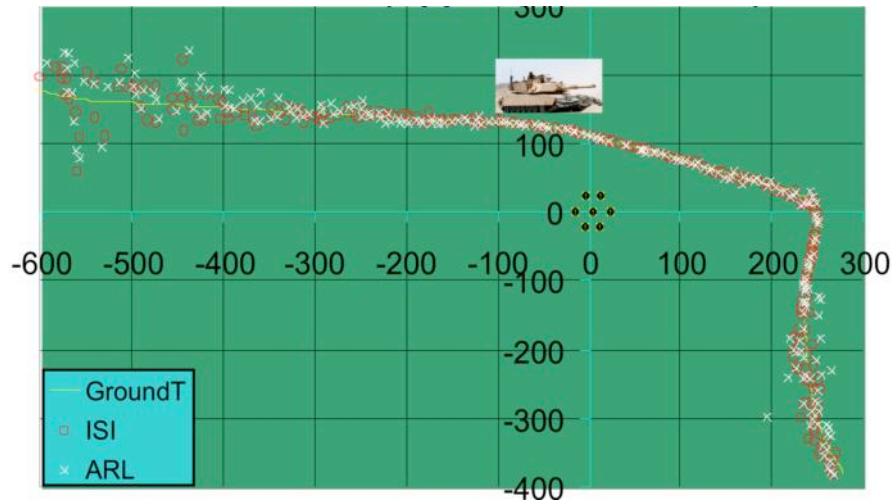
Figure 67: ISI and ARL LOB Versus Ground Truth

4.2.4 Line of Bearing Performance on HIDRA

The power results reported above were derived from simulation using JouleTrack developed at MIT. However, that tool doesn't capture the entire system-level energy picture. ISI ported a version of the ARL baseline algorithm and the optimized ISI algorithm to HIDRA and did extensive power analysis in the lab. The next four charts summarize the results. Of significant interest is that although the processing overhead does indeed virtually disappear with the ISI optimized algorithm, this results in about a 2X total system improvement because of the way that HIDRA was architected for data acquisition. In essence, the CPU is idling for most of the data acquisition window and consuming energy needlessly. A truly power aware microsensor architecture needs finer power control of all aspects of the system.

53
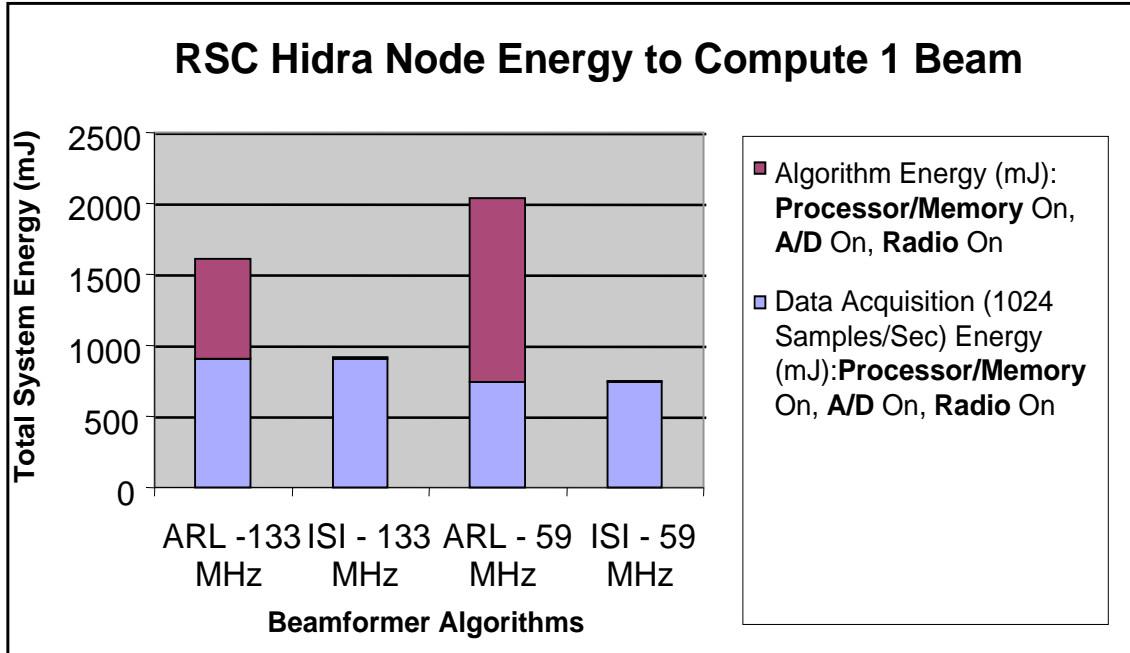
Figure 68: HIDRA Node Energy to Compute 1 Bearing
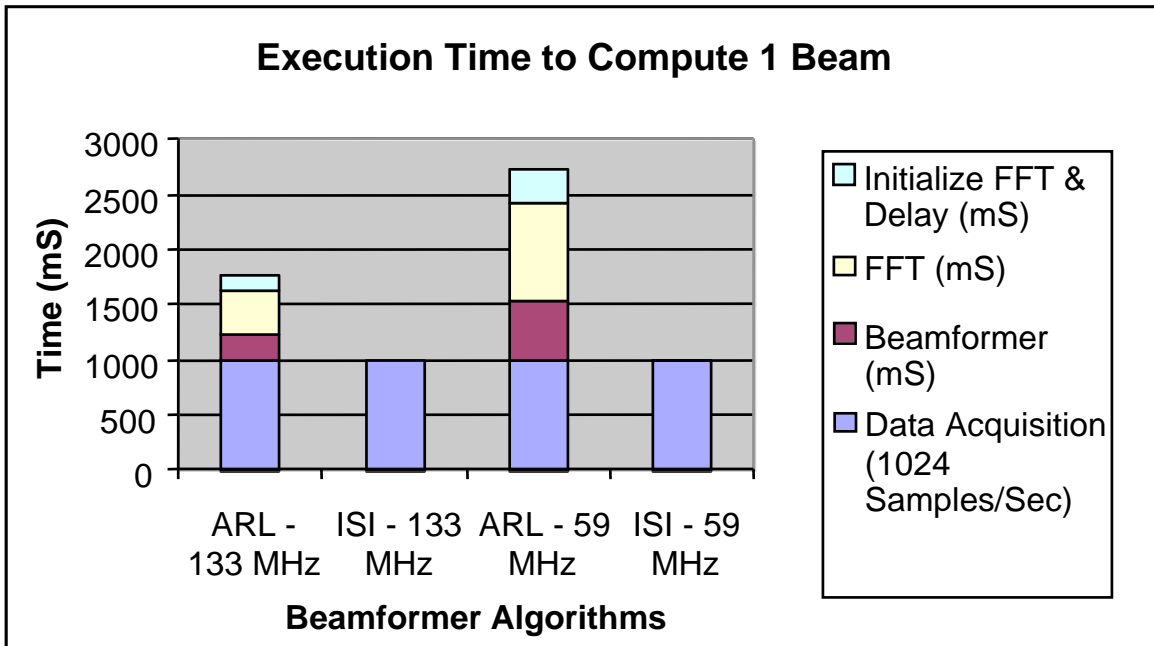
Figure 69: HIDRA Execution Time to Compute 1 Bearing
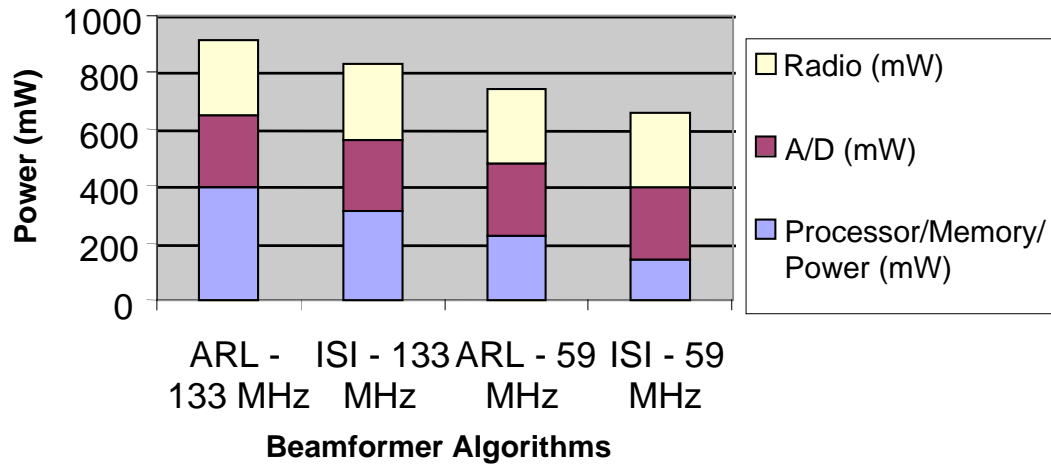
Figure 70: Average Power per Module to Calculate 1 Bearing
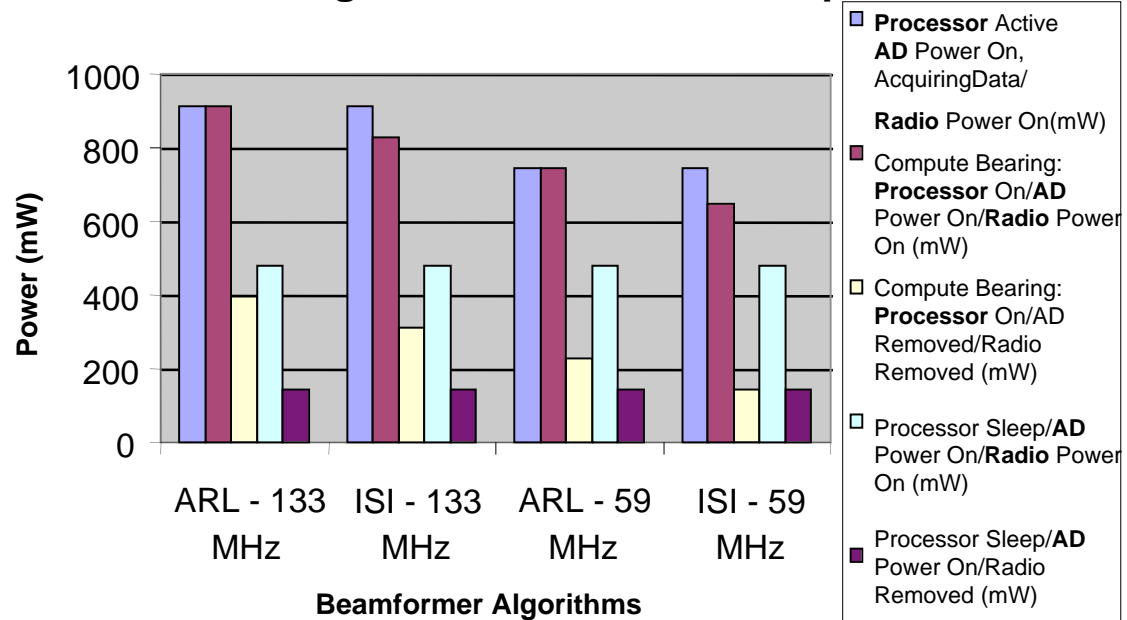
Figure 71: HIDRA Power Breakdown by Module

4.2.5 Spesuti Island Field Test

In April 2003, the PADS team participated in a field experiment on Spesuti Island at Aberdeen Proving Ground, which was organized by BAE SYSTEMS for ARL. This date was earlier than expected to be ready with a working field system, but the team was able to field two instrumented Rockwell HIDRA nodes to gather three-microphone acoustic array data for lab experiments. At this experiment, ISI also fielded two prototype IPAQ video nodes for imagery collection. The team collected 14 vehicle runs of tracked and wheeled vehicles, totaling about 8 hours. BAE SYSTEMS provided GPS ground truth. In support of this experiment, Rockwell Scientific made several modifications, in both software and in hardware, to the HiDRA wireless sensor system. Rockewll fabricated a power-supply board capable of dynamic voltage scaling for use in conjunction with software-selectable processor-core clock frequency scaling to enable power-savings modes during low computational demanding periods. In addition to the design of the power-supply board, a sensor cross-talk issue was debugged and corrected to enable multi-channel sampling on the HiDRA nodes. For this field test, ISI used new packaging that included an EMI enclosure (with shielding up to 20 GHz) and environmentally sealed connectors.
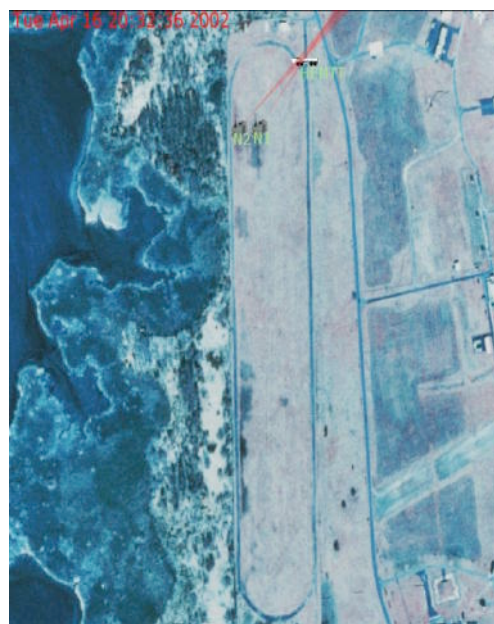
4.3   Laplacian Pyramid Image Compression

ISI developed multi-resolution image compaction algorithms capable of compressing images without loss. Lossy compression isn't appropriate because it produces image artifacts that would effect automated target recognition applications. The Laplacian Pyramid format theoretically allows an image to be transmitted with incrementally increased spatial resolution to enable remote processing of images at the resolution determined the receiver. The concept is that the image is successively down sampled by two and subtracted from the previous layer to form a pyramid. Averaging blocks of pixels during down sampling is a Gaussian blur, the difference is Laplacian.
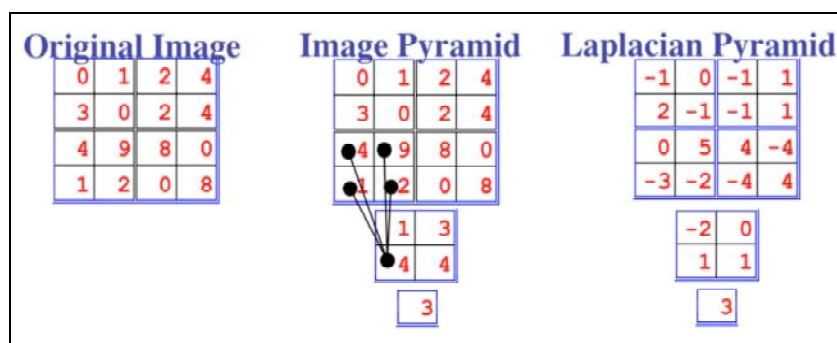
Figure 72: Laplacian Image Coding

Figure 73: Laplacian Pyramid Image

The Laplacian Pyramid method is described as follows in Figure 74. Performance results are compared in Figure 75.

Figure 74: The Laplacian Pyramid Method

Image	Orig	$\frac{B+C}{2}$	$\frac{A+B+C}{3}$	B+C-A	$\frac{3(B+C)-2A}{4}$	Pyramid
Lena	7.59	4.91	5.07	5.17	4.90	4.74
Boats	7.19	5.21	5.36	5.42	5.17	5.04
Gold hill	7.48	5.01	5.21	5.16	4.92	5.03
Airplane	6.71	4.36	4.57	4.41	4.23	4.39
Bridge	7.67	5.99	6.17	6.20	5.95	6.06

(Table title: **Estimated Average bits per pixel**)

Figure 75: Laplacian Pyramid Results

5   DELIVERABLES SUMMARY

5.1.1 Task 1: Architectural Approaches

1. Instrumentation board for research platform (RP) with RSC modules.

   - Complete – A small four-channel power instrumentation board has been developed. The module-level isolation board for RSC nodes is also complete.

2. Fine grain instrumented processor module for RP.

   - Complete – We have returned to using the Rockwell HiDRA platform and have wire-modified some nodes to enable power monitoring.

3. Land Warrior streaming multimedia support.

   - Complete.

4. FPGA radio prototypes with basic communication modules (modem only).

   - Complete.

5. FPGA radio with RF amplifier and b/w adaptive analog interface.

   FPGA radio has been completed. RF amplifier and b/w adaptive analog interface deliverable has been abandoned as result of reduction in funding.

6. FPGA radio prototypes with run-time reconfiguration support.

   Complete. Hardware for runtime reconfiguration is available in prototype.

7. Deployable platform (DP) with power aware control "knobs"/monitors.

   This deliverable has been abandoned because of a reduction in funding.

8. DP with advanced RSC processor board and technology from RP.

   This deliverable has been abandoned because of a reduction in funding.

9. RP with sensor-triggered activation to enable ultra-low power sleep mode, and technology integrated from RP [RSC/ISI: FY03/Q3].

   This deliverable has been abandoned because of a reduction in funding.

5.1.2 Task 2: Middleware, Tools, and Techniques

1. P-A scheduling in RTOS lab demonstration on RP

   Done – demonstrated on two research platforms: iPAQ, XScale.

2. P-A scheduling in RTOS field demonstration on DP.

   This deliverable has been abandoned because of a reduction in funding.

3. P-A resource management lab demonstration on RP.

   Done – Demonstrated on two research platforms: iPAQ, XScale

4. P-A resource management field demonstration on DP.

   This deliverable has been abandoned because of a reduction in funding.

5. RTOS power management simulation tool evaluation.

Done – we selected PARSEC simulation language

6. RTOS power management simulation tool demonstration.

   Done – simulator with multiple scheduling strategies available

7. RTOS power management simulation tool release.

   Done – already released to UCI.

8. Integration of SensIT P-A protocols into PAC/C.

   Done – Initial integration done on iPaq with 802.11 radios.

9. Basic hybrid simulator with gateway to RSC nodes.

   Done – SensorSim released.

10. Data collection during SensIT field demonstration using software instrumented RSC node software.

    This deliverable has been abandoned because of a reduction in funding.

11. Power models of RSC sensor nodes, protocols, and network traffic.

    Models are incorporated in the released version of SensorSim.

12. Benchmark scenarios based on RSC nodes and SensIT field data.

    This deliverable has been abandoned because of a reduction in funding.

13. Hybrid simulation/emulation framework release.

    Initial version of SensorSim already released, and is in use by many groups

14. Tool for power-aware RTOS kernel synthesis with static scheduling.

    This deliverable has been abandoned because of a reduction in funding.

15. Synthesis tool with support for dynamically scheduled RTOS kernel.

    This deliverable has been abandoned because of a reduction in funding.

16. P-A network resource allocation simulation demonstration.

    Complete.

17. P-A network resource allocation on DP, field demonstration.

    This deliverable has been abandoned because of a reduction in funding.

18. Algorithms for P-A network migration of H/W and S/W functions, demonstration on network of RP nodes with attached FPGA.

    This deliverable has been abandoned because of a reduction in funding.

19. Field demonstration of network migration using beam forming and multiresolution sensor processing algorithms.

    This deliverable has been abandoned because of a reduction in funding.

5.1.3 Task 3: Algorithms

1. Algorithm demonstration code in C for compressed image transmission with incremental resolution based on Laplacian Pyramid.

   Complete.

2. Multi-resolution acoustic beamforming code and demonstration.

   Complete. ISI has added this code to the ARL distribution CD.

3. Multiresolution image target classifier code based on neural networks.

   This deliverable has been abandoned because of a reduction in funding.

4. Multi-resolution, hierarchical sensor cueing and processing algorithm including acoustic and/or low-res imaging sensor cueing simulator demonstration.

   This deliverable has been abandoned because of a reduction in funding.

5. Directed high-resolution multi-look image classification/validation demo.

   This deliverable has been abandoned because of a reduction in funding.

6 PERSONNEL

6.1.1 USC Information Sciences Institute Personnel

Robert Parker	Director, ISI-E
Brian Schott	Project Leader
Carl Worth	Researcher
Doe-Wan Kim	Researcher
Dong-In Kang	Researcher
Joe Czarnaski	Researcher
Ron Riley	Researcher
Sukjae Cho	Graduate Research Assistant

6.1.2 UCLA Personnel

Mani Srivastava	Faculty
Rajesh Gupta	Faculty
Cristiano Ligieri	Graduate Student
Curt Schurgers	Graduate Student
Jin-seong Jeong	Graduate Student
Paleologos Spanos	Graduate Student
Pavan Kumar	Research Assistant
Ravindra Jejurikar	Graduate Student
Sung Park	Graduate Student
Vijay Raghunathan	Graduate Student

6.1.3 Rockwell Scientific Company Personnel

Charles Chien	Program PI
Igor Elgorriaga	Program PI
Jim DeMarchi	Software Engineer
Maxim Pedyash	Technical Manager
Timothy Chow	Software Engineer
Victor Lin	Researcher
Yefim Poberezhskiy	Senior Researcher

7   PUBLICATIONS

[1] C. Schurgers, V. Tsiatsis, and M.B. Srivastava, "STEM: Topology management for energy efficient sensor networks," IEEE Aerospace Conference, March 2001.

[2] A. Savvides, S. Park, and M. Srivastava, "On modeling networks of wireless micro-sensors," Proceedings of ACM SIGMETRICS 2001, June 2001.

[3] S. Park, A. Savvides, and M. Srivastava, "Battery capacity measurement and analysis using lithium coin cell battery," Proceedings of the ACM International Symposium on Low Power Electronics and Design (ISLPED), August 2001.

[4] V. Tsiatsis, S. Zimbeck, and M. Srivastava, "Architecture strategies for energy efficient packet forwarding in wireless sensor networks," Proceedings of the ACM International Symposium on Low Power Electronics and Design (ISLPED), August 2001.

[5] C. Schurgers, O. Aberthorne, and M. Srivastava, "Modulation scaling for energy aware communication systems," Proceedings of the ACM International Symposium on Low Power Electronics and Design (ISLPED), August 2001.

[6] C. Schurgers, and M. Srivastava, "Energy efficient routing in wireless sensor networks," Proceedings of MILCOM 2001, October 2001.

[7] C. Schurgers, V. Raghunathan, and M. Srivastava, "Modulation Scaling for Real-Time Energy Aware Packet Scheduling," Proceedings of IEEE Globecom, November 2001.

[8] V. Raghunathan, P. Spanos, and M. Srivastava, "Adaptive power-fidelity in energy aware wireless embedded systems," Proceedings of the IEEE Real-Time Systems Symposium, December 2001.

[9] S. Park, A. Savvides, and M. Srivastava, "Simulating networks of wireless sensors," Proceedings of the 2001 Winter Simulation Conference (WSC 2001), December 2001.

[10]   C. Schurgers, and M.B. Srivastava, "Energy Efficient Wireless Scheduling: Adaptive Loading in Time," Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'02), March 2002. Accepted.

[11]   V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-aware wireless sensor networks", IEEE Signal Processing (special issue on collaborative signal processing), March 2002.

[12]   Vijay Raghunathan, Saurabh Ganeriwal, Curt Schurgers, Mani B. Srivastava, "E2WFQ: An Energy Efficient Fair Scheduling Policy for Wireless Systems," International Symposium on Low Power Electronics and Design (ISLPED'02), Monterey, CA, August 12-14, 2002.

[13]   Ronald A. Riley, Jr., Sohil B. Thakkar, Joseph P. Czarnaski, and Brian Schott, "Power-Aware Acoustic Processing," Military Sensing Symposium: Battlefield Acoustic and Seismic Systems Conference, Columbia, MD, September 23-25, 2002.

[14]   C. Schurgers, V. Raghunathan, and M. B. Srivastava, "Power Management for Energy Aware Communication Systems", accepted for publication in ACM Transactions on Embedded Computing Systems (special issue on power aware embedded computing).

[15]   V. Raghunathan, C. Pereira, M. B. Srivastava, and R. Gupta, "Energy Aware Wireless Systems with Adaptive Power-Fidelity Tradeoffs", submitted to IEEE Transactions on VLSI Systems.

[16]   V. Raghunathan, S. Ganeriwal, C. Schurgers, and M. B. Srivastava, "Energy Efficient Wireless Packet Scheduling and Fair Queuing", submitted to ACM Transactions on Embedded Computing Systems (special issue on networked embedded computing).

8   LIST OF ACRONYMS

ARL – Army Research Labs
ASK – Amplitude Shift Keying
CODEC – Encoder / Decoder
CPU – Central Processing Unit
DARPA – Defense Advanced Research Projects Agency
DSP – Digital Signal Processor
FEC – Forward Error Correction
FFT – Fast Fourier Transform
FPGA – Field Programmable Gate Array
HAL – Hardware Abstraction Layer
ISI – Information Sciences Institute
LOB – Line of Bearing
MAC – Media Access Control
MIT – Massachusetts Institute of Technology
OOK – On/Off Keying
OS – Operating System
P-A – Power Aware
PAC/C – Power Aware Computing and Communications
PCMCIA – Personal Computer Memory Card International Association
PSK – Phase Shift Keying
QAM – Quadrature Amplitude Modulation
RSC – Rockwell Scientific Company (formerly Rockwell Science Center)
RTOS – Real Time Operating System
STEM – Sparse Topology and Energy Management
TDMA – Time Domain Multiple Access
TI – Texas Instruments
UCLA – University of California, Los Angeles
USC – University of Southern California
VLSI – Very Large Scale Integrated

9 LIST OF ADDENDA

[1] C. Schurgers, V. Tsiatsis, and M.B. Srivastava, "STEM: Topology management for energy efficient sensor networks," IEEE Aerospace Conference, March 2001.

[2] A. Savvides, S. Park, and M. Srivastava, "On modeling networks of wireless micro-sensors," Proceedings of ACM SIGMETRICS 2001, June 2001.

[3] S. Park, A. Savvides, and M. Srivastava, "Battery capacity measurement and analysis using lithium coin cell battery," Proceedings of the ACM International Symposium on Low Power Electronics and Design (ISLPED), August 2001.

[4] V. Tsiatsis, S. Zimbeck, and M. Srivastava, "Architecture strategies for energy efficient packet forwarding in wireless sensor networks," Proceedings of the ACM International Symposium on Low Power Electronics and Design (ISLPED), August 2001.

[5] C. Schurgers, O. Aberthorne, and M. Srivastava, "Modulation scaling for energy aware communication systems," Proceedings of the ACM International Symposium on Low Power Electronics and Design (ISLPED), August 2001.

[6] C. Schurgers, and M. Srivastava, "Energy efficient routing in wireless sensor networks," Proceedings of MILCOM 2001, October 2001.

[7] C. Schurgers, V. Raghunathan, and M. Srivastava, " Modulation Scaling for Real-Time Energy Aware Packet Scheduling," Proceedings of IEEE Globecom, November 2001.

[8] V. Raghunathan, P. Spanos, and M. Srivastava, "Adaptive power-fidelity in energy aware wireless embedded systems," Proceedings of the IEEE Real-Time Systems Symposium, December 2001.

[9] S. Park, A. Savvides, and M. Srivastava, "Simulating networks of wireless sensors," Proceedings of the 2001 Winter Simulation Conference (WSC 2001), December 2001.

[10] C. Schurgers, and M.B. Srivastava, "Energy Efficient Wireless Scheduling: Adaptive Loading in Time," Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'02), March 2002. Accepted.

[11] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-aware wireless sensor networks", IEEE Signal Processing (special issue on collaborative signal processing), March 2002.

[12] Vijay Raghunathan, Saurabh Ganeriwal, Curt Schurgers, Mani B. Srivastava, "E2WFQ: An Energy Efficient Fair Scheduling Policy for Wireless Systems," International Symposium on Low Power Electronics and Design (ISLPED'02), Monterey, CA, August 12-14, 2002.

[13] Ronald A. Riley, Jr., Sohil B. Thakkar, Joseph P. Czarnaski, and Brian Schott, "Power-Aware Acoustic Processing," Military Sensing Symposium: Battlefield Acoustic and Seismic Systems Conference, Columbia, MD, September 23-25, 2002.

[14] C. Schurgers, V. Raghunathan, and M. B. Srivastava, "Power Management for Energy Aware Communication Systems", accepted for publication in ACM Transactions on Embedded Computing Systems (special issue on power aware embedded computing).

[15]   V. Raghunathan, C. Pereira, M. B. Srivastava, and R. Gupta, "Energy Aware Wireless Systems with Adaptive Power-Fidelity Tradeoffs", submitted to IEEE Transactions on VLSI Systems.

[16]   V. Raghunathan, S. Ganeriwal, C. Schurgers, and M. B. Srivastava, "Energy Efficient Wireless Packet Scheduling and Fair Queuing", submitted to ACM Transactions on Embedded Computing Systems (special issue on networked embedded computing).

STEM: Topology Management for Energy Efficient Sensor Networks [2]

Curt Schurgers
University of California, Los Angeles
Eng. IV Bldg., UCLA-EE
Los Angeles, CA 90095
1-310-206-4465
curts@ee.ucla.edu

Vlasios Tsiatsis
University of California, Los Angeles
Eng. IV Bldg., UCLA-EE
Los Angeles, CA 90095
1-310-825-7707
tsiatsis@ee.ucla.edu

Mani B. Srivastava
University of California, Los Angeles
Eng. IV Bldg., UCLA-EE
Los Angeles, CA 90095
1-310-267-2098
mbs@ee.ucla.edu

Abstract—In wireless sensor networks, where energy efficiency is the key design challenge, the energy consumption is typically dominated by the node's communication subsystem. It can only be reduced significantly by transitioning the embedded radio to a sleep state, at which point the node essentially retracts from the network topology. Existing topology management schemes have focused on cleverly selecting which nodes can turn off their radio, without sacrificing the capacity of the network. We propose a new technique, called Sparse Topology and Energy Management (STEM), that dramatically improves the network lifetime by exploiting the fact that most of the time, the network is only sensing its environment waiting for an event to happen. By alleviating the restriction of network capacity preservation, we can trade off extensive energy savings for an increased latency to set up a multi-hop path. We will also show how STEM integrates efficiently with existing topology management techniques.

TABLE OF CONTENTS

1. INTRODUCTION

Sensor Networks

Sensor nodes are autonomous devices equipped with heavily integrated sensing, processing, and communication capabilities [1][2]. When these nodes are networked together in an ad-hoc fashion, they form a sensor network. The nodes gather data via their sensors, process it locally or coordinate amongst neighbors and forward the information to the user or, in general, a data sink. Due to the node's limited transmission range, this forwarding mostly involves using multi-hop paths through other nodes [3]. It is important to point out that a node in the network has essentially two different tasks: (1) sensing its environment and processing the information, and (2) forwarding traffic as an intermediate relay in the multi-hop path.

Such sensor networks find applicability in wildlife observation, smart office buildings, and applications such as battlefield or disaster area monitoring. They are also considered to establish sensor grids on distant planetary bodies, relaying information to the interplanetary Internet. In addition, future large-scale networks of resource limited satellites are likely to be governed by similar principles and can benefit from the design methodologies developed for sensor networks.

The major design challenge for this type of networks is to increase their operational lifetime as much as possible, despite the limited energy supply of each node [1][2][3]. Indeed, to provide unobtrusive operation, sensor nodes are miniature devices and, as a result, operate on a tiny, non-replaceable battery. Energy efficiency is therefore the critical design constraint.

In terms of energy consumption, the wireless exchange of data between nodes strongly dominates other node functions such as sensing and processing [1][3][4]. Moreover, the radio consumes almost as much energy in receive and idle mode as it does in transmit mode [4]. Significant energy savings are only obtainable by putting the node in sleep mode, essentially disconnecting it from the network and changing the topology. This has severe repercussions, as sleeping nodes can no longer function as relays in multi-hop paths.

Topology Management

The goal of topology management is to coordinate the sleep transitions of all the nodes, while ensuring adequate network connectivity, such that data can be forwarded efficiently to

the data sink. Existing topology management schemes try to do just that: they remove redundancy in the network topology while trying to conserve the data communication capacity [5][6]. Due to this restriction of sacrificing as little of the forwarding capacity as possible, the gains of these schemes are relatively modest, even for extremely dense networks. The underlying reasoning is that they implicitly assume the network has data to forward, which we refer to as being in the *'transfer state'*.

However, most of the time, the sensor network is only monitoring its environment, waiting for an event to happen. For a large subset of sensor net applications, no data needs to be forwarded to the data sink in this *'monitoring state'*.

Consider for example a sensor network that is designed to detect brush fires. It has to remain operational for months or years, while only sensing if a fire has started. Once a fire is detected, this information should be forwarded to the user quickly. Even when we want to track how the fire spreads, it probably suffices for the network to remain up only for an additional week or so. It is clear that although the transfer state should be energy efficient, it is far more important for the monitoring state to be ultra-low power, as the network resides in this state most of the time. Similar observations hold for applications such as surveillance of battlefields, machine failures, room occupancy, or other reactive scenarios, where the user needs to be informed once a condition is satisfied.

Of course, different parts of the network could be in monitoring or transfer state, so, strictly speaking, the 'state' is more a property of the locality of node, rather than the entire network. We also note that the network probably needs to transition to the transfer state periodically to exchange network management and maintenance messages [1].

Nevertheless, these sensor networks often spend the vast majority of time in the monitoring state. It is therefore critical to optimize the network's energy efficiency in this state as much as possible, beyond what is accomplished by existing topology management techniques.

We acknowledge that sensor networks could also be designed to periodically send updates to the data sink, or, in general, reside in the monitoring state much less frequent. In this case, the technique presented in this paper is expected to be much less useful. Yet, we foresee that the majority of sensor network applications and scenarios would have significant periods without data forwarding activity, and as such greatly benefit from the topology management technique we will present here.

Our Contributions

We observe that in the monitoring state, which we expect to be predominant, the requirement of capacity preservation is no longer pertinent. Instead, nodes only need the ability to wake up neighbors to perform coordinated sensing or set up a path, with a reasonable latency. As such, the network topology can be much sparser, and nodes spend more time sleeping.

In principle, the communication capacity could be reduced to virtually zero, by turning off the radios of all nodes (*i.e.*, putting them in the sleep mode). Note that the sensors and processor can be on at that time, since they are much less power hungry than the communication subsystem. As soon as events are detected, however, nodes need to be woken up quickly to set up the multi-hop communication path to the data sink. This requires nodes to communicate with each other, but this is only possible if they have their radio turned on. We obviously have two contradictory requirements here: on the one hand, nodes should be in sleep mode as often as possible when they are in the monitoring state, yet they should receive requests of other nodes to return to the more active transfer state.

In this paper, we propose a new topology management scheme, called ***STEM (Sparse Topology and Energy Management)***. It trades off energy consumption in the monitoring state, versus latency of switching back to the transfer state. The resulting energy savings have a significant impact on the network lifetime, which is extended in addition to and beyond existing approaches.

Prior Work

For sensor networks, two alternative routing approaches have been considered: flat multi-hop and clustering. Although STEM is applicable to both of them, we mainly focus on flat multi-hop routing [3][7][8]. For clustered approaches [9], which are possibly hierarchical, our scheme can be used to reduce the energy of the cluster heads, although the gains are expected to be less dramatic here.

Recently, topology management techniques, called SPAN [5] and GAF [6], have been proposed for flat multi-hop routing. They operate on the assumption that the network capacity needs to be preserved. As a result, the energy consumption is approximately the same whether the network is in the transfer or monitoring state, as no distinction is made between them. In contrast, STEM dramatically improves the energy efficiency in the monitoring state, far beyond what is achieved by SPAN and GAF alone, which can still be used in the transfer state. We can thus claim that STEM is in a way orthogonal to these existing techniques.

In SPAN [5], a limited set of nodes forms a multi-hop forwarding backbone, which tries to preserve the original capacity of the underlying ad-hoc network. Other nodes transition to sleep states more frequently, as they no longer carry the burden of forwarding data of other nodes. To balance out energy consumption, the backbone functionality is rotated between nodes, and as such there is a strong

interaction with the routing layer. Unlike SPAN, STEM does not try to conserve capacity, resulting in greater energy savings, and also does not impact routing.

Geographic Adaptive Fidelity (GAF) [6] exploits the fact that nearby nodes can perfectly and transparently replace each other in the routing topology. The sensor network is subdivided into small grids, such that nodes in the same grid are equivalent from a routing perspective. At each point in time, only one node in each grid is active, while the others are in the energy-saving sleep mode. Substantial energy gains are, however, only achieved in very dense networks. We will discuss this issue further on in this paper, when we integrate STEM with GAF.

An approach that is closely related to STEM is the use of a separate paging channel to wake up nodes that have turned of their main radio [10]. However, the paging channel radio cannot be put in the sleep mode for obvious reasons. This approach thus critically assumes that the paging radio is much lower power than the one used for regular data communications. It is yet unclear if such radio can be designed. STEM basically emulates the behavior of a paging channel, by having a radio with a low duty cycle radio, instead of a radio with low power consumption.

2. SPARSE TOPOLOGY MANAGEMENT

Basic Concept

In the application scenarios we consider in this paper, the sensor network is in the monitoring state the vast majority of its lifetime. Ideally, we would like to only turn on the sensors and some preprocessing circuitry. When a possible event is detected, the main processor is woken up to analyze the data in more detail. The radio, which is normally turned off, is only woken up if the processor decides that the information needs to be forwarded to the data sink.

Now, the problem is that the radio of the next hop in the path to the data sink is still turned off, if it did not detect that same event. As a solution, each node periodically turns on its radio for a short time to listen if someone wants to communicate with it. The node that wants to communicate, the *'initiator node'*, sends out a beacon with the ID of the node it is trying to wake up, called the *'target node'*. In fact, this can be viewed as the initiator node attempting to activate the link between itself and the target node. As soon as the target node receives this beacon, it responds to the initiator node and both keep their radio on at this point. If the packet needs to be relayed further, the target node will become the initiator node for the next hop and the process is repeated.

Dual Frequency Setup

Once both nodes that make up a link have their radio on, the link is active, and can be used for subsequent packets. In order for actual data transmissions not to interfere with the wakeup protocol, we propose to send them in different frequency bands using a separate radio in each band. Sensor nodes developed by Sensoria Corporation [11], for example, are already equipped with a dual radio.

Figure 1 shows the proposed radio setup. The wakeup messages, which were discussed in the subsection above, are transmitted by the radio operating in frequency band f_1. We refer to these communications as occurring in the *'wakeup plane'*. Once the initiator node has successfully notified the target node, both nodes turn on their radio that operates in frequency band f_2. The actual data packets are transmitted in this band, or what we call the *'data plane'*.
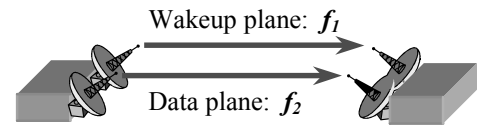
Figure 1 – Radio setup of a sensor node
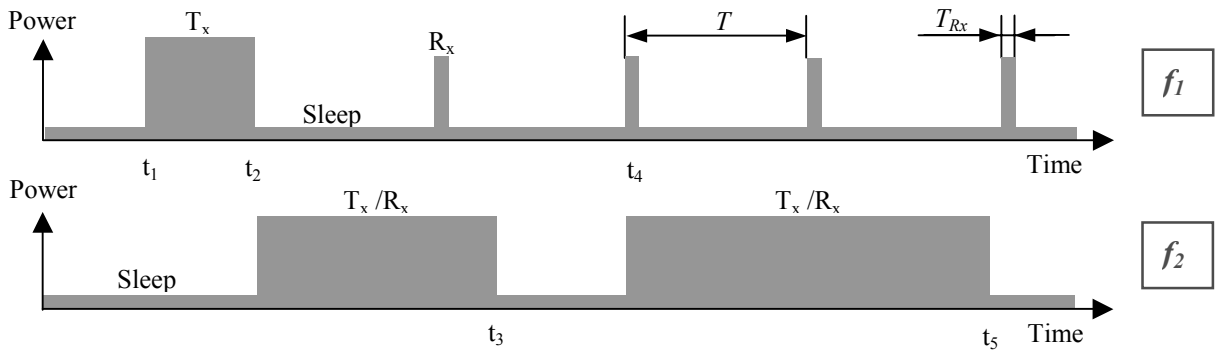
Figure 2 – State transitions of STEM for a particular node

69

STEM Operation

Figure 2 presents an example of typical radio mode transitions for one particular node in the network. Some representative power numbers for the different modes are summarized in Table 1. These numbers correspond to a 2.4 Kbps low-power RFM radio using OOK modulation, with an approximate transmit range of 20 meters [4].

Table 1. Radio power characterization

Radio mode	Power consumption (mW)
Transmit (T_x)	14.88
Receive (R_x)	12.50
Idle	12.36
Sleep	0.016

At time t_1, the node wants to wake up one of its neighbors and thus becomes an initiator. It starts sending beacon packets on frequency f_1, until it receives a response from the target node, which happens at time t_2. At this moment, the radio in frequency band f_2 is turned on for regular data transmissions. Note that at the same time, the radio in band f_1 still wakes up periodically from its sleep state to listen if any nodes want to contact it. After the data transmissions have ended (e.g. at the end of a predetermined stream of packets, after a timeout, etc.), the node turns its radio in band f_2 off again. At time t_4, it receives a beacon from another initiator node while listing in the f_1 band. The node responds to the initiator and turns its radio on again in band f_2.
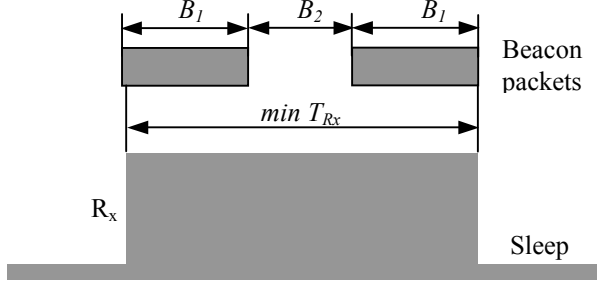
Figure 3 – Radio on-time in the wakeup plane

In order for the target node to receive at least one beacon, it needs to turn on its radio for a sufficiently long time, denoted as T_{Rx}. Figure 3 illustrates the worst-case situation where the radio is turned on just too late to receive the first beacon. In order to receive the second beacon, T_{Rx} should be at least as long as twice the transmit time B_1 of a beacon packet, plus the inter-beacon spacing B_2 that is required to allow the target node to respond.

3. THEORETICAL ANALYSIS

Setup Latency

Before simulating our protocol, we first develop a theoretical model of the system performance. We define the **setup latency T_S of a link** as the interval from the time the initiator starts sending out beacons, to the time the target node has responded to the beacon. Typically the target and originator node are not synchronized, which means that the beacon sending process starts at a random point in the cycle of the target node. As a result, the start of the first beacon is distributed uniformly random in interval T. Figure 4 shows the values of T_S, normalized versus $B_{1+2} = B_1 + B_2$, for different start times of the beacon sending process.

It is clear that T_S only takes on integer multiples of B_{1+2}, as this is the time it takes to send a beacon and receive the response to it. For the region that is labeled i in Figure 4, the setup latency is equal to $i \cdot B_{1+2}$, since beacon i is the first one to fall entirely within the interval of length T_{Rx} when the target node's radio is on. The probability of being in region i is equal to the length of that region divided by T. As a result, for $T > T_{Rx}$, the statistics of T_S can be derived from Figure 4 as:

$$\begin{cases} P(T_S = B_{1+2}) = \dfrac{T_{Rx} - B_1}{T} \\[2mm] P(T_S = k \bullet B_{1+2}) = \dfrac{B_{1+2}}{T} \qquad k = 2...K \\[2mm] P(T_S = (K+1) \bullet B_{1+2}) = \dfrac{T - K \bullet B_{1+2} - T_{Rx} + B_1 + B_{1+2}}{T} \end{cases} \qquad (1)$$
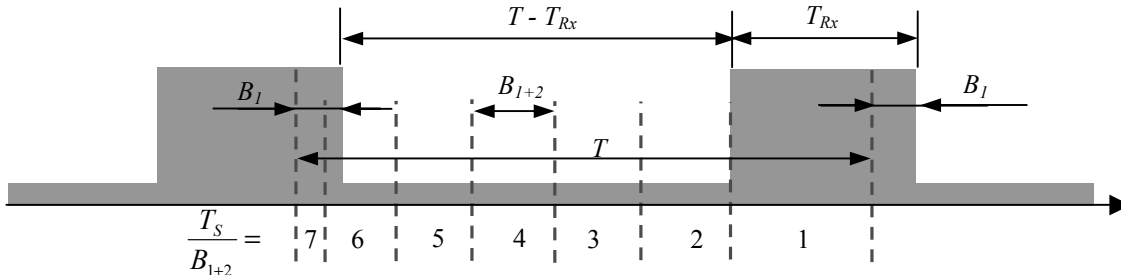
Figure 4 – Analysis of the setup latency

$$K = \left\lfloor \frac{T - T_{Rx} + B_1 + B_{1+2}}{B_{1+2}} \right\rfloor \qquad (2)$$

Based on these equations, we calculate the average setup latency $\overline{T}_S$ for a link. To simplify the expressions, we select T_{Rx} equal to its minimum value (see Figure 3):

$$T_{Rx} = B_{1+2} + B_1 \qquad (3)$$

In this case, (1)-(2) reduce to:

$$\begin{cases} P(T_S = k \bullet B_{1+2}) = \dfrac{B_{1+2}}{T} & k = 1...K \\[2mm] P(T_S = (K+1) \bullet B_{1+2}) = \dfrac{T - K \bullet B_{1+2}}{T} \end{cases} \qquad (4)$$

$$K = \left\lfloor \frac{T}{B_{1+2}} \right\rfloor = \frac{T}{B_{1+2}} - \delta \qquad (5)$$

The average setup latency per hop can be derived from (4) as being equal to (6), where δ is defined in (5).

$$\overline{T}_S = \frac{T + B_{1+2}}{2} + \frac{B_{1+2}^2}{2 \bullet T} \bullet \delta \bullet (1 - \delta) \qquad (6)$$

If T is an integer multiple of B_{1+2}, this expression simplifies to:

$$\overline{T}_S = \frac{T + B_{1+2}}{2} \qquad (7)$$

Equations (6) and (7) are valid on condition that $T > T_{Rx}$. For the special case when there is no sleep period, $T = T_{Rx}$ and the average setup delay is equal to:

$$\overline{T}_S = B_{1+2} \qquad (8)$$

Energy Savings

The total energy consumed by a node during a time interval t can be broken up into two components, one for each frequency band.

$$E_{node} = E_{wakeup} + E_{transfer} \qquad (9)$$

Equation (10) details the energy consumption in the wakeup plane. The first term accounts for the listening cycle, where P_{node} is given by (11). In this equation P^0_{node} is a combination of idle and receive power. Since both are very similar, see Table 1, we can approximate P^0_{node} by P_{idle}. The second term in (10) represents the energy consumption of transmitting beacon and response packets (P_{setup} is thus a combination of transmit, receive and idle power).

$$E_{wakeup} = P_{node} \bullet (t - t_{setup}) + P_{setup} \bullet t_{setup} \qquad (10)$$

$$P_{node} = \frac{P_{sleep} \bullet (T - T_{Rx}) + P^0_{node} \bullet T_{Rx}}{T} \qquad (11)$$

The energy consumption in the transfer plane is given by (12). In this equation, t_{data} is the total time the radio is turned on in the transfer plane for communicating data. As a result, P_{data} contains contributions of packet transmission, packet reception and idle power.

$$E_{transfer} = P_{sleep} \bullet (t - t_{data}) + P_{data} \bullet t_{data} \qquad (12)$$

Without topology management, the total energy would be equal to (13). Although P_{data} also contains contributions of P_{idle}, we have chosen to split up the energy consumption in analogy with (12) for ease of comparison. The main difference is that the radio is never in the energy-efficient sleep state here.

$$E_{node}^{original} = P_{idle} \bullet (t - t_{data}) + P_{data} \bullet t_{data} \qquad (13)$$

The gain in terms of energy obtained by using STEM is the difference between (13) and (9):

$$\begin{aligned} . E_{node} &= E_{node}^{original} - E_{node} \\ &= (P_{idle} - P_{sleep} - P_{node\bullet}) \bullet t - (P_{idle} - P_{sleep}) \bullet t_{data} \\ &\quad - (P_{setup} - P_{node\bullet}) \bullet t_{setup} \end{aligned} \qquad (14)$$

Since we consider scenarios where the node is in the monitoring state most of the time, we can roughly disregard t_{data} and t_{setup}. By ignoring the minute power of the sleep state and substituting P^0_{node} in (11) by P_{idle}, we approximate (14) as:

$$. E_{node} \quad P_{idle} \bullet t \bullet (1 - \frac{T_{Rx}}{T}) \qquad (15)$$

Furthermore, by also ignoring t_{data} in (13), we can reasonably approximate the relative gain in terms of energy as:

$$. E_{node} = \frac{. E_{node}}{E_{node}^{original}} \quad 1 - \frac{T_{Rx}}{T} \qquad (16)$$

From (6) and (16), we can derive the general relationship between the setup latency and the relative energy gain for a node. For the special case where T is an integer multiple of B_{1+2}, as in (7), this relationship is given by:

$$. E_{node} = 1 - \frac{T_{Rx}}{\left(2 \bullet \overline{T}_S - B_{1+2}\right)} \qquad (17)$$

Since the node has a finite battery capacity, these energy savings directly correspond to the same relative increase in the lifetime of a node, which ultimately results in a prolonged lifetime of the sensor network.

71

4. PERFORMANCE EVALUATION

Simulation Setup

In this section, we verify our algorithm through simulations, which were written on the Parsec platform, an event-driven parallel simulation language [12]. We distribute N nodes randomly over a square field of size L x L and each of them has a transmission range R.

For a uniform network density, the probability $Q(n)$ for a node to have n neighbors in a network of N nodes is given by the binomial distribution of (18), when edge effects are ignored. In this equation, Q_R is the probability of a node being in the transmission range of a particular node, given by (19). We use the symbol Q in this paper for probabilities, to avoid confusion with power (denoted by P).

$$Q(n) = Q_R^n \bullet (1 - Q_R)^{N-1-n} \bullet \binom{N-1}{n} \qquad (18)$$

$$Q_R = \frac{\pi R^2}{L^2} \qquad (19)$$

For large values of N, tending to infinity, this binomial distribution converges towards the Poisson distribution (20) [13]. The network connectivity is thus only a function of the average number of neighbors of a node, denoted by parameter μ.

$$Q(n) = \frac{\mu^n}{n!} \bullet e^{-\mu} \qquad (20)$$

$$\mu = \frac{N}{L^2} \bullet \pi R^2 \qquad (21)$$

Since the traffic communication patterns depend solely on the network connectivity, we only have to consider μ and not N, R and L separately. We have verified this statement through simulations, and therefore can characterize a uniform network density by the single parameter μ.

In our simulations, we have chosen $R = 20$ m, which corresponds to the numbers in Table 1. The area of the sensor network is such that for $N = 100$, we have $\mu = 20$. Furthermore, our setup includes a CSMA-type MAC, similar to the DCF of 802.11. Table 2 lists the other simulation settings, where L_{beacon} and $L_{response}$ are the sizes (including MAC and PHY header) of the beacon and the response packets respectively.

Table 2. Simulation settings

R	20 m	R_b	2.4 Kbps
L	79.27 m	B_{1+2}	150 ms
L_{beacon}	144 bits	T_{Rx}	225 ms
$L_{response}$	144 bits		

The node closest to the top left corner detects an event and sends 20 information packets of 1040 bits to the data sink with an inter-packet spacing of 16 seconds. This process will therefore take about $t^* = 320$ seconds. The data sink is the sensor node located closest to the bottom right corner of the field. We have observed that the average path length is between 6 and 7 hops. All reported results are averaged over 100 simulation runs.

Simulation Results

Figure 5 shows the average setup latency per hop as a function of the wakeup period T. The dashed curve with the markers is obtained via simulations, while the top solid curve corresponds to (6). There is a constant offset, which is due to the fact that the transmission time of a beacon and response packet is actually 120 ms, while the beacon period B_{1+2} was chosen conservatively to be 150 ms. The actual setup latency is thus comprised of a number of B_{1+2} periods, plus the time to transmit a beacon and receive the response, which is about 30 ms less than what is calculated theoretically in (6). From Figure 5, we observe that if we correct (6) by subtracting 30 ms, the correspondence to simulations is indeed very close.
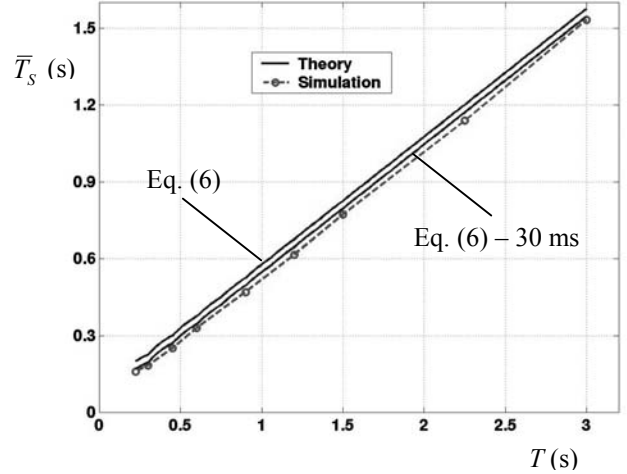
Figure 5 – Average setup latency

In Figure 6, the total energy is plotted versus the normalized observation interval t/t^*. As a basis for comparison, we included the curve for a scheme without topology management, which corresponds to (13). In this case, there is only one radio, which can never be turned off. The other dashed curves represent the performance for STEM with different values of T. The theoretical results, plotted using solid lines, are obtained by multiplying the curve without topology management by $(1 - E)$, see (16).

For all values of t, the same number of packets is sent, meaning that the duration of the transfer state is kept constant, and is approximately equal to t^*. When t increases,

the monitoring state becomes more predominant. As a result, t_{data} and t_{setup} in (10), (12), (14) are negligible for large t, such that the simulated values start approaching the theoretical ones. We observe that STEM results in energy savings when $t > 2 \cdot t^*$, which means that the network should reside in the monitoring state 50% or more of the time.
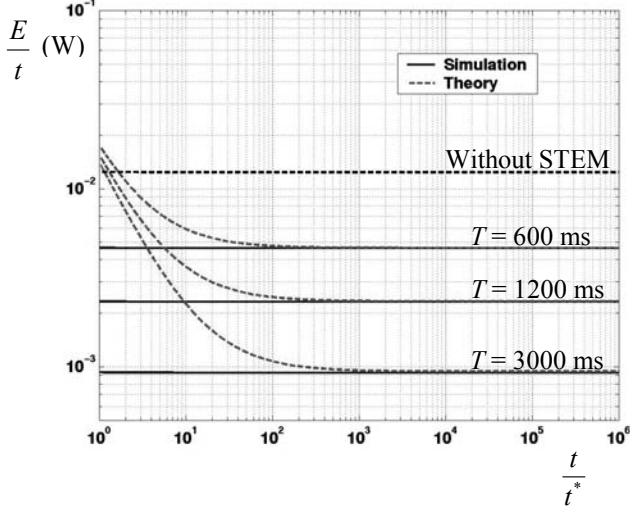
Figure 6 – Relative energy savings versus the total observation interval t

Figure 7 explicitly shows the tradeoff between energy savings and setup latency. The solid theoretical curves are obtained from (16) and (6), with and without the correction that was introduced in Figure 5. We have plotted the simulated results for values of the different observation period t.
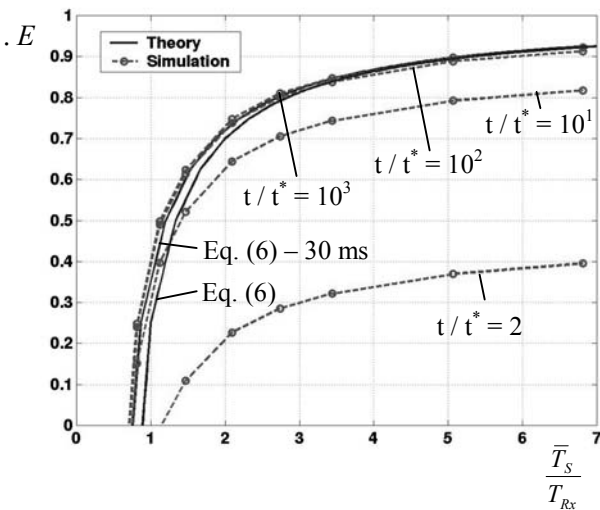
Figure 7 – Simulated energy – setup latency tradeoff

For large t, the simulated performance converges to the theoretical one. This corresponds to a regime where the monitoring state heavily dominates the transfer state, which is the focus of this work. We note, however, that STEM can also be valuable if outside this regime (*i.e.*, for smaller values of t), although the gains are much less pronounced in this case.

5. COMBINING STEM AND GAF

As mentioned in the introduction, existing topology management schemes, such as GAF and SPAN, coordinate the radio sleep and wakeup cycles while ensuring adequate communication capacity. STEM can be viewed as being orthogonal to these schemes, and additional gain is achieved by considering combinations STEM-GAF or STEM-SPAN. In this work, we specifically focus on the interaction between STEM and GAF.

GAF Behavior

In this subsection, we discuss plain GAF, *i.e.*, without STEM. The GAF algorithm is based on a division of the sensor network in a number of virtual grids of size r by r, see Figure 8. The value of r is chosen such that all nodes in a grid are equivalent from a routing perspective [6]. This means that any two nodes in adjacent grids should be able to communicate with each other. By investigating the worst-case node locations depicted in Figure 8, we can calculate that r should satisfy (22) [6].

$$r = \frac{R}{\sqrt{5}} \qquad (22)$$
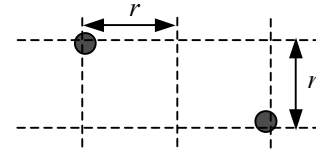
Figure 8 – GAF grid structure

The average number of nodes in a grid, M, is given by (23). By combining this with (22), we can see that M should satisfy (24). The average number of nodes in a grid is thus fairly low. Even if r satisfies (22) with equality, which we assume to hold for the remainder of this paper, M is smaller than 2 for densities of $. = 31$. To put this into perspective, $. = 31$ corresponds to a topology where each node has 31 neighbors on average.

$$M = \frac{N}{L^2} \bullet r^2 \qquad (23)$$

$$M = \frac{\cdot}{5\pi} \qquad (24)$$

Since all nodes in a grid are equivalent from a routing perspective, we can use this redundancy to increase the

network lifetime. GAF only keeps one node awake in each grid, while the other nodes put their radio in the sleep mode. To balance out the energy consumption, the burden of traffic forwarding is rotated between nodes. For simplicity, we ignore the unavoidable time overlap of this process associated with handoff. If there are m nodes in a grid, the node will (ideally) only turn its radio on $1/m^{th}$ of the time and therefore will last m times longer. However, equation (24) shows that the redundancy is rather low on average, even for fairly dense networks.

When distributing nodes over the sensor field, some grids will not contain any nodes at all. We use $.$ to denote the fraction of used grids, *i.e.*, which have at least one node. As a result, the average number of nodes in the used grids is equal to M', given by:

$$M' = \frac{M}{.} \qquad (25)$$

The average power consumption of a node using GAF, $\overline{P}_{node}^{GAF}$, is equal to (26). In this equation, P_{on} is the power consumption of a node if GAF would not be used. It thus contains contributions of receive, idle and transmit mode, as the node would never turn its radio off. With GAF, in each grid only one node at a time has its radio turned on, so the total power consumption of a grid, P_{grid}, is virtually equal to P_{on} (neglecting the sleep power of the nodes that have their radio turned off). Since M' nodes share the duties in a grid equally, the power consumption of a node is $1/M'$ that of the grid, as in (26).

$$\overline{P}_{node}^{GAF} = \frac{P_{on}}{M'} = \frac{P_{grid}}{M'} \qquad (26)$$

The average relative gain in energy for a node is thus given by:

$$. E_{node} = \frac{P_{on} \bullet t - P_{node}^{GAF} \bullet t}{P_{on} \bullet t} = 1 - \frac{1}{M'} \qquad (27)$$

Alternatively, we see that the lifetime of each node in the grid is increased with the same factor M'. As a result, the average lifetime of a grid, $\bar{t}_{grid}$, *i.e.*, the time that at least one node in the grid is still alive, is given by (28), where t_{node} is the lifetime of a node without GAF. *We can essentially view a grid as being a 'virtual node', composed of M' actual nodes.*

$$\bar{t}_{grid} = t_{node} \bullet M' \qquad (28)$$

Note that $\overline{P}_{node}^{GAF}$ and $\bar{t}_{grid}$, which are averages over all grids, only depend on M' and not on the exact distribution of nodes in the used grids! Of course, the variance of both the node power and the grid lifetime depends on the distribution. If we would have full control over the network

deployment, we could make sure that every used grid has exactly M' nodes, which minimizes the power and lifetime variance.

For the special case of a random node distribution, we now calculate the statistics exactly. The probability $Q(m)$ of having a grid with m nodes is given by (29). The derivation is analogous that the one leading to (20).

$$Q(m) = \frac{M^m}{m!} \bullet e^{-M} \qquad (29)$$

In this case, the fraction $.$ of used grids is equal to:

$$. = 1 - Q(0) = 1 - e^{-M} \qquad (30)$$

The probability of having m nodes in a used grid is given by:

$$Q(m|m=1) = \frac{Q(m)}{Q(m=1)} = \frac{M^m}{m!} \bullet \frac{e^{-M}}{1 - e^{-M}} \qquad (31)$$

We also know that the probability that power of a node is equal to $1/m^{th}$ of that in a grid, is the same as the probability of a node being in a grid with m nodes:

$$Q(P_{node}^{GAF} = \frac{P_{grid}}{m}) = \frac{m \bullet Q(m)}{M} = \frac{M^{m-1}}{(m-1)!} \bullet e^{-M} \qquad (32)$$

Alternatively, equation (33) gives the probability that the lifetime of a grid is m times that of an individual node.

$$Q(t_{grid} = t_{node} \bullet m) = Q(m|m=1) = \frac{M^m}{m!} \bullet \frac{e^{-M}}{1 - e^{-M}} \qquad (33)$$

We can verify from (32) and (33) that the average values of P_{node}^{GAF} and t_{grid} are indeed equal to (26) and (27) respectively.

Interaction of STEM and GAF

As mentioned before, GAF essentially places one virtual node in each grid, and the physical nodes alternatively perform the functionalities of that virtual node. From this perspective, combining GAF with STEM is straightforward by envisioning the virtual node as running STEM. In real life, nodes alternate between sleep and active states, as governed by GAF. The one active node in the grid, runs STEM in the same way as described in section 2. The only difference is that now the routing protocol needs to address virtual nodes (or grids) instead of real nodes.

This insight allows us to directly modify the expressions of section 3 to similar ones for the combination of STEM-GAF. In particular, (16) becomes (34), where the statistics of m are given by (32).

$$. E_{node} = 1 - \frac{1}{m} \bullet \frac{T_{Rx}}{T} \qquad (34)$$

74

When considering the average behavior over all grids, we get:

$$. E_{node} = 1 - \frac{1}{M\ni} \bullet \frac{T_{Rx}}{T} \qquad (35)$$

If T is an integer multiple of B_{1+2}, we can combine (35) with (7) to obtain the following tradeoff between energy savings and setup latency:

$$. E_{node} = 1 - \frac{1}{M\ni} \bullet \frac{T_{Rx}}{\left(2 \bullet \overline{T}_S - B_{1+2}\right)} \qquad (36)$$

Figure 9 plots this tradeoff for different values of M'. As argued before, these curves are independent of the exact node distribution, but only depend on M'.
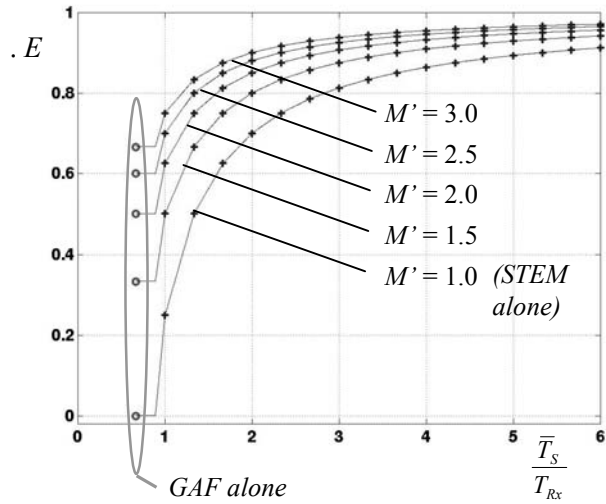
Figure 9 – Theoretical energy – setup delay tradeoff

The solid curves are based on (6) and (35). They therefore represent the behavior as averaged over the different grids, for any $T > T_{Rx}$. On these curves, the '+' markers are points obtained from (36), where T is an integer multiple of B_{1+2}.

The circles mark the limiting case where the wakeup-plane radio is always on ($T = T_{Rx}$). This case corresponds to a traditional paging channel setup, where a separate paging radio is used to wake up the main data radio [10]. Of course, this only makes sense if the paging radio is substantially more energy efficient than the main one. By looking at (27) and (35), we notice that in this case the energy savings are also the same as those of pure GAF, without STEM, which corresponds to intuition. The circles can therefore be viewed as the (energy) behavior of GAF as well, although, strictly speaking, the setup latency does not have any true meaning here.
By comparing (16) and (35), we note that the curve with M' = 1 can also be viewed as representing the case of STEM without GAF, where essentially no node redundancy is exploited. So, besides the combination of STEM-GAF, figure 9 also shows the behavior of GAF without STEM (circles) and of STEM without GAF (curve with $M' = 1$).

We notice that by allowing more setup latency, the energy savings can be increased considerably beyond what is achievable by GAF alone. For a uniform node deployment, the values of M' in Figure 9 translate to M and . as given by (21), (23) and (25). Table 3 lists the values of these parameters for the curves of Figure 9.

Table 3. Density mapping for a uniform node distribution

M'	M	.
1.0	0	0
1.5	0.87	13.7
2.0	1.59	25.0
2.5	2.22	35.0
3.0	2.82	44.3

Note that for moderate node densities (. < 25), the average redundancy in a used grid is fairly low. As a result, GAF alone only results in moderate energy saving, below 34%. On the other hand, by incorporating STEM, we can achieve savings of more than 93%! In other words, the energy is reduced to 66% of the original value by GAF and to a mere 7% by also using STEM. The penalty is of course an increased setup delay.

6. CONCLUSIONS

In this paper, we have introduced STEM, a topology management technique that trades off power savings versus path setup latency in sensor networks. It emulates a paging channel by having a separate radio operating at a lower duty cycle. Upon receiving a wakeup message, it turns on the primary radio, which takes care of the regular data transmissions.

Our topology management is specifically geared towards those scenarios where the network spends most of its time waiting for events to happen, without forwarding traffic. STEM leverages the fact that, while awaiting events, the network capacity can be heavily reduced, resulting in energy savings.

We have shown that STEM integrates directly with other topology management schemes such as GAF, and results in energy savings above and beyond these existing techniques. Compared to a network without topology management, a combination of GAF and STEM can reduce the energy consumption to a mere 7%. Alternatively, this results in a node lifetime increase of a factor 14! However, these

benefits come at the cost of increased setup latency, which is linearly proportional to the number of hops in the multi-hop path. It will depend on the specific applications, how much latency is allowed, and therefore how far the energy consumption can be scaled down.

Analyzing the interaction of STEM and SPAN is a topic of future research. Another issue worth investigating is how power control strategies can be incorporated into topology management.

7. ACKNOWLEDGEMENTS

REFERENCES

[1] K. Sohrabi, J. Gao, V. Ailawadhi, G. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications Magazine*, Vol.7, No.5, pp. 16-27, Oct. 2000.

[2] L. Clare, G. Pottie, J. Agre, "Self-organizing distributed sensor networks," *SPIE - The International Society for Optical Engineering*, Orlando, FL, pp. 229-237, April 1999.

[3] D. Estrin, R. Govindan, "Next century challenges: scalable coordination in sensor networks," *MobiCom 1999*, Seattle, WA, pp. 263-270, August 1999.

[4] A. Savvides, C.-C. Han, M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," *MobiCom 2001*, Rome, Italy, pp. 166 – 179, July 2001.

[5] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *MobiCom 2001*, Rome, Italy, pp. 70-84, July 2001.

[6] Y. Xu, J. Heidemann, D. Estrin, "Geography-informed energy conservation for ad hoc routing," *MobiCom 2001*, Rome, Italy, pp. 70-84, July 2001.

[7] J.-H. Chang, L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," *INFOCOM 2000*, Tel Aviv, Israel, pp. 22-31, March 2000.

[8] J. Rabaey, J. Ammer, J.L. da Silva, D. Patel, "PicoRadio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes," *IEEE Computer Society Workshop on VLSI 2000*, Orlando, FL, pp. 9-12, April 2000.

[9] W. Rabiner Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *HICSS 2000*, Maui, HI, Jan. 2000.

[10] S. Kumar, "DARPA sensor information technology (SensIT)," *http://www.darpa.mil/ito/research/sensit/*.

[11] Sensoria Corporation, *http://www.sensoria.com/*.

[12] R. Bagrodia, R. Meyer, M. Takai, Y.A. Chan, X. Zeng, J. Marting, H.Y. Song, "Parsec: a parallel simulation environment for complex systems," *Computer*, Vol.31, No.10, pp. 77-85, October 1998.

[13] M. Yacoub, *Foundations of Mobile Radio Engineering*, CRC Press, 1993.

Curt Schurgers *received his MSEE summa cum laude from the Katholieke Universiteit Leuven (KUL), Belgium, in 1997. From 1997 to 1999, he worked at IMEC (Interuniversity Micro Electronics Center, Leuven, Belgium) on memory optimization techniques for turbo codes. Currently, he is pursuing his Ph.D. degree UCLA, focusing on energy efficient communication and networking systems. Curt Schurgers has received the F.W.O., the B.A.E.F. and the UCLA fellowships in 1997, 1999 and 2000 respectively.*

Vlasios Tsiatsis *has received his BS degree from the Technical University of Crete, Chania, Greece in 1998, and his M.S. degree in EE from the University of California, Los Angeles (UCLA), in 2001. He is currently pursuing his Ph.D. degree at UCLA, researching low power protocol architectures. In 1998 and 2000, he was honored with the UCLA fellowship.*

Mani Srivastava *is an Associate Professor of Electrical Engineering at UCLA. He received his B.Tech. in EE from IIT Kanpur in India and M.S. and Ph.D. from Berkeley. From 1992 through 1996 he was a Member of Technical Staff at Bell Laboratories in Networked Computing Research. His current research interests are in mobile and wireless networked computing systems, low power systems, and sensor networks. He received the NSF CAREER award in 1997, and the President of India Gold Medal in 1985.*

On Modeling Networks of Wireless Microsensors

Andreas Savvides, Sung Park and Mani Srivastava
Electrical Engineering Department
University of California, Los Angeles
{asavvide, spark, mbs}@ee.ucla.edu

1. INTRODUCTION

Recent advances in low-power embedded processors, radios, and micro-mechanical systems (MEMs) have made possible the development of networks of wirelessly interconnected sensors. With their focus on applications requiring tight coupling with the physical world, as opposed to the personal communication focus of conventional wireless networks, these wireless sensor networks pose significantly different design, implementation, and deployment challenges. Their application-specific nature, severe resource limitations, long network life requirements, and the presence of sensors lead to interesting interplay between sensing, communications, power consumption, and topology that designers need to consider. Existing tools for modeling wireless networks focus only on the communications problem, and do no support modeling the power and sensing aspects that are essential to wireless sensor network design. In this paper, we present a set of models and techniques that are embodied in a simulation tool [1] for modeling wireless sensor networks. Our models are derived with detailed power measurements involving 2 different types of sensor nodes representing two extremes; high-end WINS nodes [2] by Rockwell and low-end experimental nodes that we have built. The WINS nodes have a StrongARM S1100 processor and a 100m-range radio and can carry a wide variety of sensors. The experimental nodes feature an AVR 90LS8535 microcontroller from Atmel and a low power radio 20m-range from RFM Monolithics and a similar to the COTS nodes from UC Berkeley [3].

To instrument sensor network scenarios in a simulation environment, more features need are also introduced. The notion of a sensing channel is used to propagate stimuli to the sensors. Target models are responsible for generating the stimuli that trigger the sensors, which in turn become communication traffic towards a central base station. All these actions affect power consumption, which directly affect the useful lifetime of the network. Since power consumption is a crucial factor we focus our study on empirical measurements of power consumption on sensor nodes that can be use to produce accurate models in a simulation environment. The sections that follow provide a brief overview of the sensor node and battery models and present the results of our power measurements.

2. SENSOR NETWORK AND NODE MODELS

A sensor network is modeled as a set of heterogeneous entities. Sensor nodes deployed over the area of interest are triggered by a certain set of stimuli that eventually result in a sensor report that is transmitted to a remote base station. Following this paradigm in a simulation environment, three main types of sensor nodes need to be created supported: 1) **target nodes** that stimulate the sensors, 2) **sensor nodes** to monitor events and 3) **user nodes** that query the sensors and are the final destination of the target reports.

The most interesting model is that of the sensor node. In addition to the communication protocol stack, this node model also includes a sensing stack that provides the interface to the sensor physical layer. The two stacks are connected together with an application layer, and together they constitute the algorithmic component of the node. To model power consumption, power models of the individual components are provided together with a battery model. As the protocols execute on the hardware, a corresponding amount of energy is depleted from the battery. Figure 1 provides an overview of the node model. This provides a flexible parametrizable model that can be applied to different sensor node architectures. The challenge in achieving an accurate sensor node model is to understand how the node consumes power.
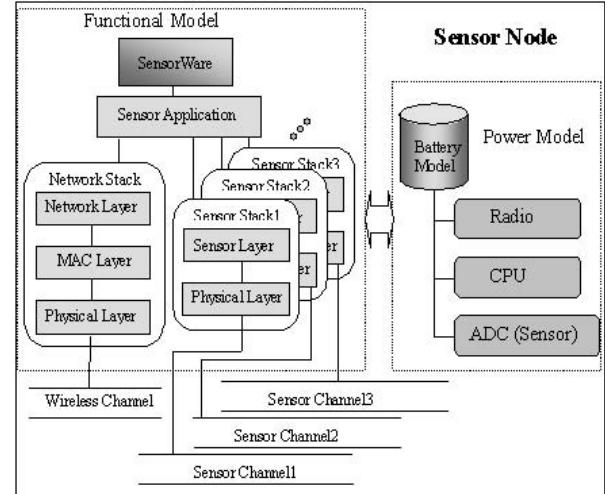
Figure 1 Sensor Node Model

3. BATTERY MODELS

3.1 Linear Battery Model

In this model, the battery is treated as linear bucket of energy. The maximum capacity of the battery is achieved regardless of what the discharge rate is. Such a model allows the user to examine the efficiency of applications by providing a simple metric of energy consumption. The remaining capacity after

Table 1 CPU Measurements for WINS and Experimental Nodes

CPU	Sensor	WINS		Experimental Node	
		Current	Power	Current	Power
Active	On	42.23mA	380mW	2.9mA	8.7mW
Sleep	On	7.0mA	64mW	1.9mA	5.9mW
Off	On	2.6mA	23.8mW	N/A	N/A
Power Down	Power Down	100μA	0.9mW	1μA	3μW

Table 2 WINS Radio Measurements

Radio Mode	Tx Power (mW)	Current Drawn (mA)	Power Consumed (mW)
Tx	0.12	43.64	395.99
Tx	0.16	43.68	396.35
Tx	0.23	43.82	397.61
Tx	0.30	43.95	398.77
Tx	0.44	44.14	400.48
Tx	0.95	45.5	412.68
Tx	1.32	45.95	416.72
Tx	1.78	45.86	424.87
Tx	2.51	47.77	433.03
Tx	3.47	48.68	441.18
Tx	10.00	59.59	538.63
Tx	13.80	63.23	571.02
Tx	19.05	68.23	615.43
Tx	27.54	73.68	663.70
Tx	36.31	79.14	711.94
Rx		41.41	375.96
Idle		38.68	351.4

Table 3 RFM Radio Measurements

Mode	Power Level	OOK Modulation				ASK Modulation			
		2.4Kbps		19.2Kbps		2.4Kbps		19.2Kbps	
	mW	mA	mW	mA	mW	mA	mW	mA	mW
Tx	0.7368	4.95	14.88	5.22	15.67	5.63	16.85	5.95	17.76
Tx	0.5506	4.63	13.96	4.86	14.62	5.27	15.80	5.63	16.85
Tx	0.3972	4.22	12.76	4.49	13.56	4.90	14.75	5.18	15.54
Tx	0.3307	4.04	12.23	4.36	13.16	4.77	14.35	5.04	15.15
Tx	0.2396	3.77	11.43	4.04	12.23	4.45	13.43	4.77	14.35
Tx	0.0979	3.13	9.54	3.40	10.35	3.81	11.56	4.08	12.36
Rx	-	4.13	12.50	4.13	12.50	4.13	12.50	4.13	12.50
Idle	-	4.08	12.36	4.08	12.36	4.08	12.36	4.08	12.36
Sleep	-	0.005	0.016	0.005	0.016	0.005	0.016	0.005	0.016

Figure 2 RFM Radio Power Comparisons

operation duration of time t_d can be expressed by following equation. Remaining capacity (in Amp*Hour) =

$$U = U' - \int_{t=t_0}^{t_0+t_d} I(t)dt$$, where U' is the previous capacity and

$I(t)$ is the instantaneous current drawn from the sensor node at time t.

3.2 Discharge Rate Dependent Model

The maximum battery capacity is very much dependent on the discharge rate or the rate at which the current is withdrawn from the battery. At high discharge rates, the battery capacity is significantly reduced. To consider this effect of discharge rate dependency, we introduce factor k which is the battery capacity efficiency factor that is determined by the discharge rate. The

definition of k is, $k = \dfrac{C_{eff}}{C_{tot}}$, where C_{eff} is the effective battery

capacity and C_{tot} is the total rated capacity of the battery with both terms expressed in unit of Ampere*hour(Ah).

3.3 Relaxation Model

Real-life batteries exhibit a general phenomenon called "relaxation". The relaxation occurs when the discharge current from the battery is cutoff or reduced after draining the battery at high discharge rate. As the discharge rate of the battery drops, the battery's cell voltage recovers, and the battery has a chance to recover the capacity lost due to the high discharge rate. The relaxation phenomenon is adapted to our battery model to simulate the behavior of real life battery.

4. POWER CHARACTERIZATION

Sensor node power consumption depends on the node's mode of operation (receive, transmit, sleep, power down). During its lifetime, a node may switch between different operational modes according to a specific task or power management scheme. By measuring the power consumption at the different operational modes accurate models can be constructed and useful insight can be obtained about the individual components of the sensor nodes.

Using an HP 1660 oscilloscope and a high precision resistor we measured the power consumption of the radio and CPU of 2 types of nodes (WINS and Experimental nodes) at different operational modes. Table 1 shows the power measurements for the CPUs on the 2 nodes. The power consumption for the WINS radio is shown in table 2. The power consumption for the RFM radio is shown in table 2 and figure 2.

These measurements show some notable trends of how power consumption is distributed in a sensor node. In both nodes, the radio consumes the most power; 50-67% of the total power consumption. Furthermore, the difference in power consumption between transmission and reception in low power radios is very small. For the WINS radio the transmission power is at most 2 times greater than reception and 1.4 times greater for the RFM radio. At some power levels, the transmit power is smaller than the receive power (figure 2).

5. REFERENCES

[1] ns-2 simulator, http://www.isi.edu/nsnam/ns

[2] Wireless Integrated Networks Systems (WINS), http://wins.rsc.rockwell.com

[3] http://www.cs.berkeley.edu/~jhill/tos/

[4] DR3000 ASH Radio Module, http://www.rfm.com/products/data/dr3000.pdf

Battery Capacity Measurement And Analysis
Using Lithium Coin Cell Battery

Sung Park, Andreas Savvides, Mani B. Srivastava
Networked and Embedded Systems Laboratory
Electrical Engineering Departments
University of California, Los Angeles
{spark, asavvide, mbs}@ee.ucla.edu

ABSTRACT

In this paper, we look at different battery capacity models that have been introduced in the literatures. These models describe the battery capacity utilization based on how the battery is discharged by the circuits that consume power. In an attempt to validate these models, we characterize a commercially available lithium coin cell battery through careful measurements of the current and the voltage output of the battery under different load profile applied by a micro sensor node. In the result, we show how the capacity of the battery is affected by the different load profile and provide analysis on whether the conventional battery models are applicable in the real world. One of the most significant finding of our work will show that DC/DC converter plays a significant role in determining the battery capacity, and that the true capacity of the battery may only be found by careful measurements.

Keywords

Embedded System, Battery, Power Estimation, Energy Estimation, DC/DC Converter, Coin Cell, Data Acquisition

1. INTRODUCTION

Proliferation of battery-powered mobile devices such as handhelds, cell phones, and pagers has given motivation to look for ways to prolong the lifetime of the battery. Furthermore, the slow improvement of the battery technology relative to the growth of power demand from these mobile devices has been fueling many studies in characterization and optimization of power usage of mobile devices so that the battery can be efficiently utilized. One of the vital pieces necessary in characterizing the power usage of the mobile system is the accurate battery model. An accurate battery model can reveal the efficiency of wireless protocols and power management schemes used in the mobile devices while an inaccurate model may tell a story far different from reality. However, this vital piece of the puzzle have remained a stumbling block for many of the electrical engineers and computer scientists due to batteries' complex technologies which involve many intricate highly non-linear electrochemical phenomenon. Moreover, many factors such as battery dimension, makeup of anode or cathode, and transport or diffusion rate of active materials, that contribute to the characteristics of the battery

have kept the battery technology in the domain of electrochemists.

In spite of these difficulties, there have been recent efforts [1],[6],[7] to generalize these complexities of the battery by modeling batteries' inherent characteristics. These models range from simple linear model to a complex model that attempts to incorporate the "relaxation" phenomenon. So far, these models, though novel in conceptual sense, lacked the validation in the real world. Moreover, studying these models does not provide the bottom line for a mobile device designer who simply wants to find out what the maximum lifetime of his or her circuit will be. Much of this is contributed by the difficulties mentioned in previous paragraph. In an attempt to overcome these difficulties, in this paper we propose a technique that can be used in characterizing the battery capacity. The technique is to carefully measure the battery's current and voltage output for the duration of the battery lifetime as the embedded board consumes power from the battery. This in turn will provide accurate measurement of the battery capacity under different load profile generated by the embedded board. Not only the technique can help estimating the battery capacity, but it can also be used to validate some of the battery models that have been proposed to see whether the outcomes foreseen by those models are accurate in reality. One primary focus of the paper is to look at the impact of the DC/DC converter by finding out whether the capacity delivered to the circuit is actually same as what was consumed from the battery. The paper is organized as follows. We provide an overview of some of the battery models in interest in section 2, and discuss how the measurement is done in section 3. We include the results and the analysis of the measurement in section 4, and we conclude the paper with section 5.

2. BATTERY MODELS

Recent efforts in modeling the battery capacity are captured in this section. These models can also be viewed as different generations of battery models with later generations incorporating additional characteristics of the battery technology. The metrics that are used to indicate the maximum capacity of the battery is in the unit of Ah (Ampere*Hour). The metric is a common method used by the battery manufacturers to specify the theoretic total capacity of the battery. Knowing the current discharge of the battery and the

total capacity in Ah, one can compute the theoretical lifetime of the battery using the equation , $T = \dfrac{C}{I}$, where T=battery lifetime, C=rated maximum battery capacity in Ah, and I=discharge current. More on this metric can be found in [5],[7].

The following subsections are brief descriptions of three battery capacity models that we consider.

2.1 Linear Model – 1st Generation

In Linear Model, the battery is treated as linear storage of current. The maximum capacity of the battery is achieved regardless of what the discharge rate is. The simple battery model allows user to see the efficiency of the user's application by providing how much capacity is consumed by the user. The remaining capacity after operation duration of time t_d can be expressed by the following equation.

$$\text{Remaining capacity (in Ah)} = C = C' - \int_{t=t_0}^{t_0+t_d} I(t)dt \text{ , Eq.(1)}$$

where C' is the previous capacity and $I(t)$ is the instantaneous current consumed by the circuit at time t. The Linear Model assumes that $I(t)$ will stay the same for the duration t_d, if the operation mode of the circuit does not change (i.e. radio switching from receiving to transmit, CPU switching from active to idle, etc..) for the duration t_d. With these assumptions equation 1 simply becomes as the following.

$$C = C' - \int_{t=t_0}^{t_0+t_d} I(t)dt = C' - I \cdot t \Big|_{t_0}^{t_0+t_d} = C' - I \cdot t_d \text{ , Eq(2)}$$

The total remaining capacity is computed whenever the discharge rate of the circuit changes. Being the most simplistic model, Linear Model falls short of portraying the behavior of a real life battery with characteristics such as rate dependent capacity and relaxation.

2.2 Discharge Rate Dependent Model – 2nd Generation

While Linear Model assumes that the maximum capacity of the battery is unaffected by the discharge rate, Discharge Rate Dependent Model considers the effect of battery discharge rate on the maximum battery capacity. In [1] [5] [15], it is shown that battery's capacity is reduced as the discharge rate increases. In order to consider the effect of discharge rate dependency, we introduce factor k which is the battery capacity efficiency factor that is determined by the discharge rate. The definition of k is,

$k = \dfrac{C_{eff}}{C_{max}}$, where C_{eff} is the effective battery capacity and C_{max}

is the maximum capacity of the battery with both terms expressed in unit of Ah. In Discharge Rate Dependent Model, the equation 1 is then transformed to the following.

$$C = k \cdot C' - I \cdot t_d \text{ , Eq.(3)}$$
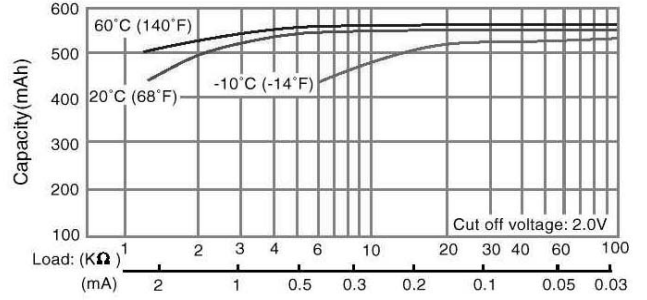
Figure 1. Capacity vs. Discharge Rate Curve for CR2354

The efficiency factor k varies with the current I and is close to one when discharge rate is low, but approaches 0 when the discharge rate becomes high. One way to find out what the k value is for different current value I is to use the table driven method introduced in [7]. With a table driven method, the factor k can be looked up from a plot similar to figure 1 which can be obtained from battery manufacturer's data sheet [9]. The figure 1 plots the battery capacity (C_{eff}) versus different discharge rate I. Using the plot, whenever the remaining capacity is computed (equation 3), the factor k can be obtained from the plot by looking at the total efficiency of the battery capacity for given current I. One shortfall of Discharge Rate Dependent Model is the fact that it does not portray the behavior of real battery by neglecting the effect of relaxation.

2.3 Relaxation Model – 3rd Generation

Real-life batteries exhibit a general phenomenon called "relaxation" explained in [1],[5],[6]. When the battery is discharged at high rate, the diffusion rate of the active ingredients through the electrolyte and electrode falls behind. If the high discharge rate is sustained, the battery reaches its end of life even though there are active materials still available. However, if the discharge current from the battery is cutoff or reduced during the discharge, the diffusion and transport rate of active materials catches up with the depletion of the materials. This phenomenon is called relaxation effect, and it gives the battery chance to recover the capacity lost at high discharge rate. For a realistic battery simulation, it's important to look at the effects of relaxation as it has effect of lengthening the lifetime of the battery. [1] introduces an analytical model which takes discharge rate as input and computes the battery voltage over the simulation duration. On the other hand, [6], [8] introduce a stochastic model, where the recovery is modeled as a change of state which occurs at a pre-set discharge rate. Obtained using the analytical model from [4], the curve in figure 2 demonstrates the effect of
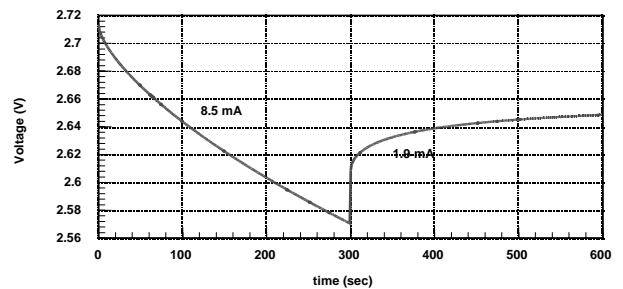
Figure 2. Recovery Effect of Relaxation Model

relaxation. In figure 2 the battery voltage recovers when the battery discharge rate is reduced to 1.9 mA after being discharged at 8.5 mA. Although the relaxation model is the most comprehensive model that closely describes the behavior a real battery, there exists considerable difficulty in implementing such model since the relaxation effect involves many electrochemical and physical properties of the battery. This is demonstrated by the analytical model from [4] which contains over 50 electrochemical and physical parameters of inputs that need to be measured separately for different types of battery.

This difficulty gives motivation for estimating the battery capacity based on a measurement based approach.

3. MEASUREMENT SETUP
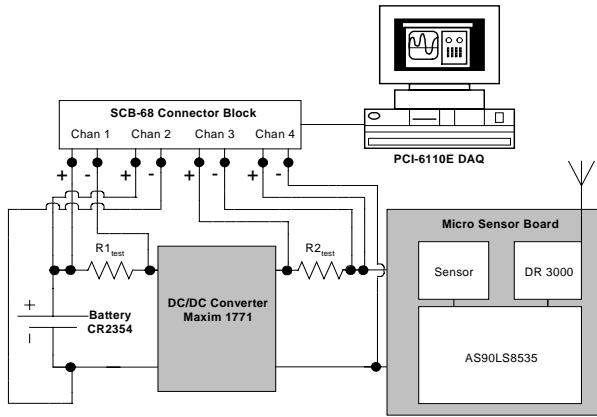
Figure 3. Battery Measurement Setup

In this paper, we propose an entirely different approach of estimating the battery capacity. The approach is to measure the current and voltage output of the battery as an embedded board consumes current out from the battery. The measurement is performed for different load profile until the battery's cutoff voltage is reached and the effective capacity of battery is computed.

Figure 3 shows the setup of the battery capacity measurement performed on a lithium coin cell, CR2354. There are total of 4 channels that monitor the current and voltage output of the battery (Chan1 and Chan2), and the current and the voltage output (Chan3 and Chan4) of DC/DC Converter. In order to measure the current, we measure the minute voltage drop across two small resistors, R1$_{test}$ and R2$_{test}$, which are 2.2 Ω high precision resistors. While Chan1 and 2 monitor how much current and power are consumed from the battery, Chan3 and 4 measures how much current and energy are actually delivered to the sensor board based on different load profiles. The values from the four channels are recorded to PC that encloses the PCI-6110E DAQ board. The measurement starts as a freshly charged (factory sealed) battery is placed on the circuit and micro sensor board is turned on. The measurement is stopped when the battery reaches the minimum voltage of 2.0V at which point DC/DC converter no longer supplies required voltage to the micro sensor board. The followings are the description of each components used in the measurement.

3.1 Micro Sensor Board

Table 1. Node States And Current Consumption at 3.3V

Operation Mode	AVR State	RFM State	Avg Current
Tx	*ON*	*TX*	12.2 mA
Rx	*ON*	*RX*	9.0 mA
Idle	*ON*	*SLEEP*	7.8 mA
Sleep	*SLEEP*	*SLEEP*	4.2 mA

The wireless sensor node we have built uses an RFM DR3000 radio module [3] and an AVR 90LS8535 microcontroller [2] from Atmel. The radio can transmit at 2 different data rates (2.4Kbps and 19.2Kbps) and supports OOK and ASK modulation. The microcontroller has 8 Kbytes of flash memory, 512 bytes of SRAM and 512 bytes of EEPROM. It is also equipped with an 8 channel 10 ADC, a programmable UART and an SPI bus. Our current implementation has a prototyping area that can support a wide variety of add on sensors such as magnetometers, thermometers, accelerometers and light sensors. This node is a variant of the widely used Smart Dust [14] nodes from UC Berkeley. The board also includes an array of ultrasound sensors that can be used to infer node location. Our main focus is the different power modes of the node that are shown in table 1.

3.2 Lithium Coin Cell Battery – CR2354 [1]

For the measurement, CR2354 lithium coin cell is used. With high energy density and relatively flat discharge characteristics, lithium batteries are widely used in mobile devices such as cellular phones, digital cameras, and PDAs. Especially for micro sensor nodes, small form factor is an essential necessity and the size of lithium coin cell batteries satisfies the stringent requirement of micro sensor nodes. Table 2 provides the manufacturer's specification of CR2354 which can be obtained from [9].

Table 2. CR2354 Specification

Output Voltage	Cutoff Voltage	Max. Capacity	Dimension Diam.xHt.	Wt
3.0 V	2.0V	560mAh	23.0mm x 5.4mm	5.9 g

3.3 DC/DC Converter (MAXIM 1771[2])

As the battery is discharged, the voltage across the battery constantly decreases. When the battery is directly connected to a VLSI circuit without any form of voltage regulator, the circuits' performance will start to degrade as the voltage across the battery decreases. Moreover, when the battery's voltage reaches the minimum input voltage required by the circuit, the circuit will stop functioning even though there may be some capacity left in the battery. It is the role of DC/DC converter to provide a constant voltage to the circuit while utilizing complete capacity of the battery. One type of DC/DC converter is called "switching regulator". A switching regulator is a circuit that uses an inductor, a transformer, or a capacitor as an energy-storage element to transfer energy from input to output in discrete packets [13]. Switching regulators can be configured to be either step up (boost) or step down (buck) or inverting with respect to the input voltage [12]. Due to their efficiency and versatility, switching regulators have been popular in battery powered mobile devices. When considering battery capacity measurement, it is important to

---

[1] Manufactured by Panasonic

[2] Manufactured by Maxim Integrated Products, Inc.

look at the operation DC/DC converters since the converters discharge the battery based on its own voltage regulator function. By doing so, DC/DC converters completely change the load profile generated by the VLSI circuit. In our measurement, we look at the both input and the output of the DC/DC converter to see how the operation DC/DC converter impacts the overall performance of the battery.

The DC/DC converter used in the measurement is MAXIM 1771 step up switching controller, which can provide a constant 3.3V from the minimum input of 2V. Maxim 1771 uses the current-limited pulse-frequency-modulation (PFM) where the charge cycle is cut-off when a predetermined peak inductor current is reached and it remains at the cut-off stage for a pre-determined fixed duration (called one-shot time constant)[13]. The effect of DC/DC Converter on the battery current and voltage output will be shown in section 4.

3.4 PCI-6110E [1]DAQ Board with SCB-68 [3] Connection Block

PCI-6110E board is a real time data acquisition board that can scan up to 4 input channels simultaneously. The board is equipped with 12bit ADC per channel with sampling rate of up to 5 Meg Samples/sec. The board uses SCB-68 connection block to connect to the actual input channel. For the most of the measurement the board was configured to scan at 5k samples/sec for each channel and some of the detailed measurements were done at 25k samples/second. Also, in order to measure the minute voltage drop across $R1_{test}$ and $R2_{test}$, the precision of the board was configured to 97.66 µV for Chan1 and Chan3. The precision of Chan2 and 4 was set at 2.44 mV. More information on the
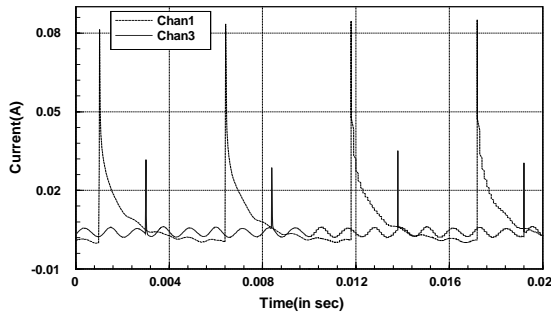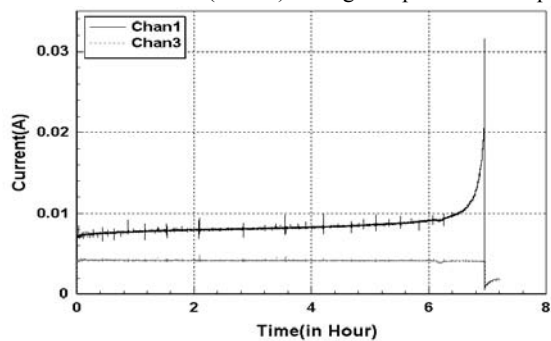
DAQ board can be found from [10], [11].

4. RESULTS

This section shows the performance of the battery by configuring the sensor board into different operation modes displayed in table 1. We look at how the battery capacity is affected by the discharge rate, DC/DC converter's voltage regulation function, and discharge profile of the sensor board. Figure 4 and 5 show snapshots (20 msec) of the current and voltage output observed from all 4 channels. One noticeable effect shown in figure 4 is that the current drawn by the sensor board (Chan3) is quite different from the current from the battery (Chan1) as the current drawn by the sensor stays around constant 4 mA while the current drawn out of the battery ranges from 80 mA to 0 mA. The current discharge pulse seen at Chan1 is the direct result of the DC/DC converters' PFM switching function which completely changes the load profile of the sensor board. This difference can also be seen from the voltage outputs shown in figure 5. The relaxation effect plays a role in the battery as the voltage across the battery (Chan2) drops when the charge cycle starts then recovers when the discharge current is cutoff. Compared to the voltage seen at Chan2, the voltage output of the DC/DC converter (Chan4) doesn't vary much as it is kept above 3.3 V.

These differences between the output seen from the battery and the output seen by the sensor board becomes clearer when the current and voltage values are observed for the complete cycle of the battery life time. Figure 6 and 7 show how the battery is utilized as the sensor board stays at sleep mode. The curves in figure 6 and 7 are smoothed by averaging the samples collected during one second interval. In figure 6 the current output from

Figure 4. Current Output from CR2354 (Chan1) and DC/DC Converter (Chan3) during Sleep Mode – Snap Shot
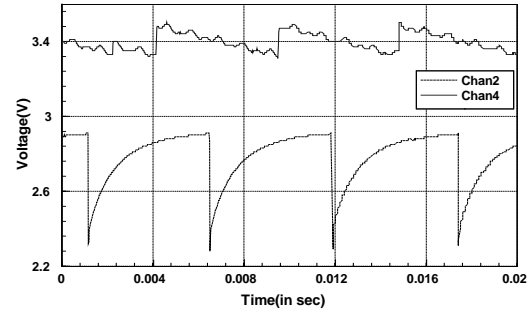
Figure 5. Voltage Output from CR2354 (Chan2) and DC/DC Converter (Chan4) during Sleep Mode – Snap Shot

Figure 6. Average Current Output from CR2354 (Chan1) and DC/DC Converter (Chan3) during Sleep Mode - Complete
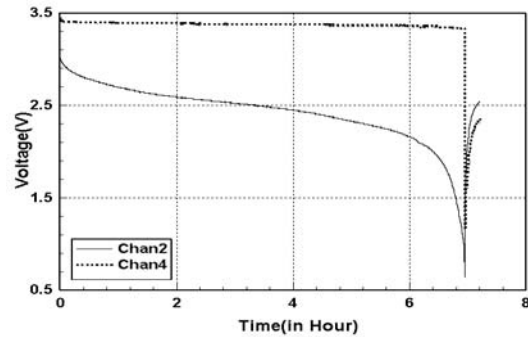
Figure 7. Average Voltage Output from CR2354 (Chan2) and DC/DC Converter (Chan4) during Sleep Mode – Complete

---

[3] Product of National Instruments Corp.

Table 3. Battery Utilization for Various Sensor Operation Modes

Operating Mode	Total Lifetime	Capacity consumed from Battery	Capacity delivered to Sensor	Energy consumed from Battery	Capacity delivered to Sensor	DC/DC converter efficiency
Tx	.33 hr	8.1 mAh	4.1 mAh	70.0 J	49.1 J	70%
Rx	.88 hr	15.8 mAh	7.9 mAh	135.2 J	96.1 J	71%
Idle	1.2 hr	18.0 mAh	9.0 mAh	154.0 J	109.0 J	71%
Sleep	7.0 hrs	59.1 mAh	28.8 mAh	505.1 J	350.7 J	69%
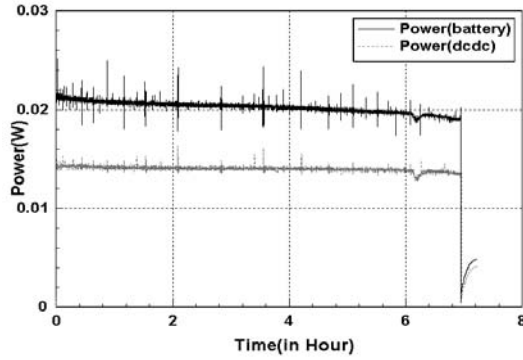
Figure 8. Power Output from CR2354 and DC/DC Converter (Chan3) during Sleep Mode

Figure 9. Current drawn from the battery during Sleep Mode and Tx Mode

Figure 10. Impact of Pulse Discharging on Battery

the battery increases as much as 32 mA when the DC/DC converter pulls more and more current out from the battery while the voltage across the battery drops (figure 7). This increase in current discharge continues until it reaches the cutoff points where DC/DC converter shuts down which occurs after approximately 7 hours. One interesting thing to notice in figure 6 is that the total capacity of 28.8 mAh (area under the curve) delivered to the sensor board is much smaller than what the DC/DC converter consumed from the battery which is 59 mAh. This difference can be explained by studying the power curves displayed in figure 8. Figure 8 plots the power consumed from the battery and the power delivered to the sensor board which can be computed by multiplying the current and the voltage plot. The decrease in the voltage across the battery (Chan2 in figure7) is matched by the increase in the current drawn from the battery (Chan1 in figure6) to give relatively constant power consumption over the course of the battery lifetime. Another factor that contributes to the difference in the current capacity is the efficiency factor of the DC/DC converter. This is demonstrated by the difference in energy (area under the curve) between the two curves shown in figure 8.

In table 3 the differences between what is consumed from the battery and what is actually delivered to the sensor board at different operating modes are listed. One trend that can be seen from the table is the characteristic that coincides with the discharge rate dependent model. Higher the discharge current, less capacity is utilized from the battery. This result comes despite the fact that the actual current discharge from the battery doesn't stay constant but fluctuates in discharge and relaxation cycles due to the PFM function implemented in DC/DC converter. Looking closely at the discharge curves in figure 9, it can be seen that when the sensor board draws a high current (in Tx mode), the battery doesn't have a chance to relax as the DC/DC converter pumps the current at high rate whereas in Sleep mode the rate of discharging cycle is much lower thus giving the battery more time to relax.

Shown in figure 10 is the effect of pulse discharging as the sensor board cycles between Rx mode and the Sleep mode. This case measures the battery capacity when a TDMA scheme is used with the sensor board which will be switching between the Rx mode and the Sleep mode within a fixed TDMA frame. For the measurement, the Rx duration is set at a fixed slot of 20 msec and the Sleep duration was varied (in multiple of 20 msec) to observe what impacts the pulse discharging has on the battery capacity. The result shows that there is 100% increase in the battery capacity utilization when every Rx slot is followed by one Sleep slot (1:1 ratio). At 1:5 ratio, there is almost 250% improvement on the battery utilization. The increase in the battery utilization is sustained with longer duration of Sleep modes but quite never reaches the battery utilization achieved for Sleep modes even at 1:20 ratio. The plot shown in figure 10 can be used in estimating the effective battery capacity of a wireless device that uses a TDMA scheme with fixed schedules.

5. CONCLUSION
In this paper we have looked at various battery models that have been introduced so far. Though these models give us qualitative insights into how battery's capacity is influenced by multiple

factors, they do not provide a simple way of finding out what the true battery capacity running in different operation modes. One approach that we have taken in this paper is based on a measurement approach, where we measure the capacity of the battery under different load profile to determine what the realized capacity of the battery is. The result shows that there are many factors such as the discharge rate, the discharge profile (constant vs. pulsing), and the DC/DC converter that govern the effective battery capacity. Especially, the role of the DC/DC converter in determining the battery capacity is a key factor that the conventional battery models don't account for. Since DC/DC converter changes the load profile generated by the sensor board based on its own voltage regulator function, we argue that the realistic battery model should consider the impact of DC/DC converter on the battery capacity. The result also suggests that a meaningful estimation of battery capacity can be achieved by measurements and not by just looking up what the manufacturer specifies. Moreover, when studying a low power design, just considering the energy number may not be sufficient as the battery lifetime has a profound dependency on how the battery is discharged. Our technique introduced in this paper provides a reasonable way of estimating the effective battery capacity and the means of justifying the low power design.

6. REFERENCES

[1] T. F. Fuller, M. Doyle, J. Newman, "Simulation and Optimization of the Dual Lithium Ion Insertion Cell," Journal of Electrochem. Soc., vol. 141, no. 4, Apr. 1994, pp. 1-10.

[2] Atmel AS90LS8535 product website, http://www.atmel.com/atmel/products/prod200.htm last accessed on 2/8/2001

[3] RFM Software Designer's Guide: http://www.rfm.com/corp/apnotes.htm last access on 2/8/2001

[4] John Newman's Website: http://www.cchem.berkeley.edu/~jsngrp/ last accessed on 2/8/2001.

[5] H.D. Linden, Handbook of Batteries, 2nd ed., McGrawHill, New York 1995.

[6] C. F. Chiasserini and R. R. Rao, "Pulsed battery discharge in communication devices," Proceedings of Mobicom 99, Seattle, August 1999

[7] T. Simunic, L. Benini, G. De Micheli, "Energy-Efficient Design of Battery-Powered Embedded Systems," Proceedings of International Symposium on Low Power Electronics and Design, pp 212-217, Piscataway, August 1999

[8] D. Panigrahi, C. Chiasserini, S. Dey, R. Rao, A. Raghunathan, and K. Lahiri, "Battery Life Estimation of Mobile Embedded System", 14th International Conference on VLSI Design, 2001

[9] Panasonic Lithium Coin Data Sheet: http://www.panasonic.com/industrial_oem/battery/battery_oem/chem/lith/lith.htm last accessed on 2/8/2001.

[10] National Instruments Corp., PCI-6110E/6111E User's Manual, April 1998 ed., Austin TX 1998

[11] National Instruments Corp., NI-DAQ User Manual for PC Compatibles, January 2000 ed., Austin TX 2000

[12] M. D. Bruce, "Step-up/Step-down converters power small portable systems", EDN, February 3 1994

[13] Maxim Integrated Products, Inc., DC-DC Converter Tutorial, http://dbserv.maxim-ic.com/tarticle/view_article.cfm?article_id=93 last access on 2/12/2001

[14] M. Kahn, R.H. Katzand, K.S.J Pister, "Mobile Networking for Smart Dust", ACM|IEEE Int. Conf. on Mobile Computing and Networking, Seattle, WA, Aug 1999

[15] M. Pedram, Q. Wu, "Battery-Powered Digital CMOS Design", Proceedings of Design, Automation and Test in Europe Conference and Exhibition, Munich, Germany, March 1999

Architecture Strategies for Energy-Efficient Packet Forwarding in Wireless Sensor Networks

Vlasios Tsiatsis, Scott A. Zimbeck, Mani B. Srivastava

Networked & Embedded Systems Laboratory, E.E. Dept, UCLA
Los Angeles, CA, 90095

{tsiatsis, szimbeck, mbs}@ee.ucla.edu

ABSTRACT

The energy-efficient communication among wireless sensor nodes determines the lifetime of a sensor network and exhibits patterns highly dependable on the sensor application and networking software. This software is responsible for processing the sensor data and disseminating the data to other nodes or a central repository. In this paper we propose a node architecture that takes advantage of both the intelligence of the radio hardware and the needs of applications to efficiently handle the packet forwarding. It exploits principles widely used in modern firewall network architectures and as our analysis shows achieves considerable energy savings.

Keywords

Energy-efficient packet forwarding, sensor networks

1. INTRODUCTION

Recently wireless ad-hoc sensor networks have gained considerable attention by the research community because of the new challenges they pose to researchers. The limited power and computational resources as well as the distinct types of data they carry and the data centric applications[2] running on them call for a different approach in constructing the overall sensor node architecture.

Figure 1 shows a simplified version of the overall sensor architecture widely adopted by researchers[6][9][11]. The major parts of a wireless microsensor system are: a) the sensor node processing subsystem running on the sensor node main CPU, b) the sensor subsystem, and c) the communication subsystem. The applications executing on a sensor node utilize all the subsystems to collect, process and receive (transmit) data from (to) other sensor nodes in the vicinity. Our proposed architecture strategies are based on the fact that the distinct communication layers shown in Figure 1 are not actually implemented in one board or device. The network communication functionality is split between the main CPU and the radio board, which are connected together by a slow serial packet link. We show later in the paper that it is because of this fact that a considerable amount of energy is wasted when packets cross the boundary between the two physical

components. A forwarded packet crosses the slow serial link twice regardless of what part (application or network layer) determines the need for forwarding (Figure 1).
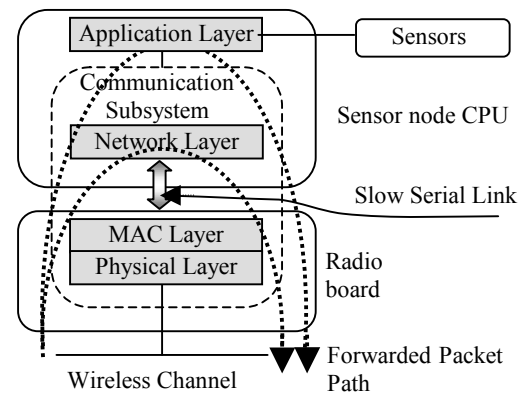
Figure 1. Typical Wireless Sensor Node Architecture

Although current radio boards have processing power in the form of a microcontroller unit (MCU) [6][9][11], this is only used for the physical and MAC layer implementation. We advocate that part of the functionality of the network layer can migrate from the main sensor CPU to the radio board in an application-defined manner. Devices that incorporate an MCU and some amount of Configurable Logic (FPGA) as the Triscend E520[10] or the Atmel FPSLIC[5], can also be the host of a limited number of network layer functions. The advantage of these devices is that several operations such as link layer destination checking, CRC checking, can be performed in the configurable logic, thus alleviating the MCU and the main node CPU. Higher level and more complex packet processing (network layer specific) can also be performed in the MCU thus allowing more sophisticated packet filtering to take place nearer to the radio hardware. With the advent of these new technology devices, protocol specific processing can be performed in two stages, namely hardware and software inside the same device.

A significant number of packets are processed in a simple manner and forwarded to the next hop. For the sake of overall delay and power consumption these packets should be processed as near to the radio hardware as possible[1]. To show this we measured the percentage of received packets (both routing and data from all the nodes) that were accepted, dropped or forwarded for a specific scenario simulated on the NS-2 network simulator. The scenario consists of a sensor terrain of 1000x1000 units inside of which 30 sensor nodes are uniformly placed. The transmission range of the radio of each node is 250 units. Two nodes are picked at random to be the source and the sink of a session of a constant-rate flow

of packets. We used IEEE 802.11 MAC, and DSR[3] as the routing protocol. From this simulation we found that 65.567% of the packets were forwarded and, 34.3% of the packets were accepted. The rest were duplicate packets that were dropped.

Sensor networks are application-specific data dissemination networks. Individual applications should have the flexibility of defining their own methods of processing and routing their data and packets. In this paper we present a network communication subsystem architecture which employs multiple levels of packet processing in order to provide: a) Ability for the communication subsystem to stop or redirect packet flow in a sensor node as low in the protocol stack as possible and b) Methods for applications to define their own routing protocols at run time.

The rest of the paper is structured as follows: Section 2 describes the proposed architecture. Section 3 presents the analysis and measurements for our prototype. Finally Section 4 concludes the paper.

2. ARCHITECTURE

Our proposed application-defined forwarding architecture has its roots in the various packet filtering architectures like the BSD Packet Filter [4]. The user specifies the packet filter as a set of packet field matching rules connected together as an expression tree with AND/OR operators. The packet field matching rules designate the field of the packet to be checked (start byte offset, length, mask), the matching value and the matching operator.

Our packet processing architecture consists of two parts: the first located on the radio board and the second on the sensor node (Figure 2). We assume that the radio board contains a MCU as the main processing component and some amount of configurable logic (FPGA). The task of the radio board part is to drop or redirect received packets that, according to the rules dictated by the applications, should not enter the sensor node. The task of the sensor node part (Router Manager) is to perform more sophisticated packet processing and packet flow demultiplexing. Each application dictates the routing rules for each part by using a filter specification explained later in the paper. In this specification it designates the assignment of the rules to each part. We assume that the application programmer has knowledge of the sensor node resources and capabilities, and as a result, s/he should assign the necessary routing rules to the appropriate parts.

2.1 Rules and Actions

Applications specify their routing protocols by defining two components: the rules against which a desired packet is matched and the action(s) taken upon a packet match. Simple rules and simple actions are specified using tuples of the general form: {byte_offset, bit_length, bit_mask, value, operator}. Rules contain either comparison or ALU operations. Using the specified tuple fields, rules perform the operation, Packet[byte_offset : bit_length] & bit_mask OPERATOR value. All rules evaluate to either true or false. Comparison rules are tuples that specify operators that test for equality, inequality or bit settings. These operators are EQ, GT, GTE, LT, LTE, and SET which perform their respective tests on packets. ALU rules use supplied tuple information to perform the following arithmetic operations on packets: ADD, SUB, MUL, DIV, AND, OR, LSH, RSH, NEG. These rules always return true. The result of the evaluation is held in an accumulator A or a register X for evaluation using the above-mentioned comparison rules.

Combining ALU rules allows filters to perform arbitrarily complex operations on packet data. Additionally, there are two operators, LD and ST that perform loads and stores to and from a packet offset, the accumulator, or the register.
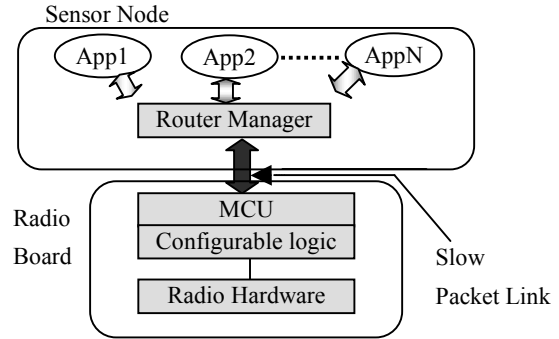
Figure 2. Two tier architecture

Actions can be of the following: terminating, and non-terminating. Terminating Actions are actions that, when encountered, stop packet filtering for the current packet. These actions are ACCEPT, DROP, and FORWARD_EX. ACCEPT is used when an application wishes to be the exclusive recipient of a matched packet. DROP simply drops the packet. FORWARD_EX (exclusive) results in the retransmission of the current packet over the radio channel. Non-terminating actions perform operations on a packet and allow subsequent rules and actions to be performed on the packet. These include modification actions, COPY, FORWARD_SH (shared), and RETURN. Modification actions allow the content of any packet to be changed. Modification is realized by using the ALU rules described above in conjunction with the LD and ST operators. COPY is similar to ACCEPT, except that the packet is allowed to continue in processing. In the same manner, FORWARD_SH allows the packet to be further filtered after being forwarded. RETURN is an action that merely marks the end of a sequence of rules and actions. Composite rules and actions are sequences of simple rules and simple actions joined by the logical operators AND/OR. An application will typically specify a number of rules and actions represented as tuples and a separate AND/OR expression tree whose leaf nodes are these specified tuples (see Figure 3). Packet processing, for a given packet and filter specification, is the evaluation of leaf-node rules and actions (tuples) on the data contained in the packet during the traversal of the application filter's expression tree. In Figure 3 we define two rule tuples namely A, B, which are connected together with AND/OR operators (*, + respectively) to construct the following composite rules: a) If rule A is false then the filter falls down to the Drop rule which drops the packet, b) If rules A and B are true then Action1 is executed.

2.2 Two-tier Packet Processing Architecture

Our two-tier architecture is presented in Figure 4. We assume that the radio hardware is a simple receiver/transmitter that is able to identify the start of packet and notify the configurable logic (CL) that this event has occurred. It is also capable of streaming the packet bits into the CL. Otherwise all the necessary functionality for packet framing within a continuous bitstream is implemented in the CL. Moreover, upon a packet transmission the CL should be able to notify the radio hardware that a packet is ready for

transmission and next be able to stream the packet bits into the radio hardware.

The block diagram of hardware packet processing component residing in the CL is depicted in Figure 4.

```
tuple_t Tuples[] = {
  {A_offset, A_length, A_mask, A_value, A_operator}, // A
  {B_offset, B_length, B_mask, B_value, B_operator}, // B
  {0, 0, 0, Action1_value, OP_Action1}, // Action1
  {0, 0, 0, 0, OP_DROP} // Drop
};
filter_tree_t App_Example_filter {
  "A*(B*Action1)+Drop;" //expression
};
```

Figure 3: Example of a C-code application-defined routing filter specification

The hardware architecture consists of the control unit (CU), several packet filed matchers (PFM), a receive (RX_FIFO) and a transmit FIFO (TX_FIFO), a CRC module and a Boolean vector. The CU is responsible for all the control signaling between the MCU and all the hardware modules residing in the configurable logic. This includes the programming of the PFMs and any packet transmissions from the TX_ FIFO. PFMs, are used to match incoming packet fields against fixed values or other packet fields. All the parameters for field matching (start bit, length, mask, fixed value, operator) are downloaded to the PFMs by the CU at the programming stage for the Configurable Logic. The CRC check module performs Cyclic Redundancy Check on every received packet and compares it with the corresponding value stored in the packet. The Boolean vector stores the outcome of the comparison operation performed in each PFM. The supported comparison operators are equal, not equal, greater than, less than, greater than or equal, less than or equal.
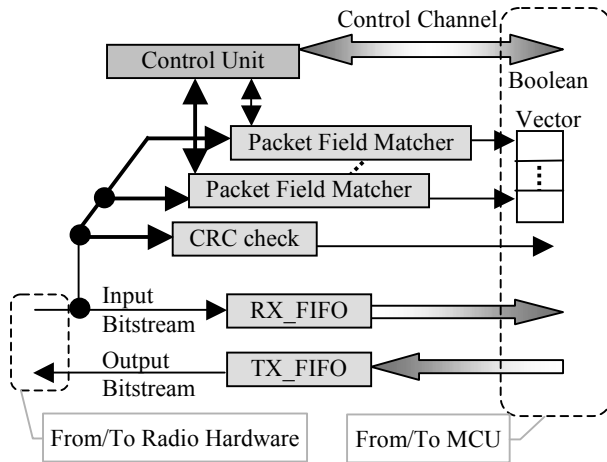
Figure 4. Configurable Logic Architecture

When signaled by the radio hardware that a new packet is being streamed in the CL, the control unit enables all the PFMs to start scanning the bitstream and perform the field matching according to their programmed values. At the same time the packet is being stored in the RX_FIFO queue and the CRC is being calculated. Shortly after the packet reception is finished, the control unit signals the MCU that a new packet is ready in the RX_FIFO queue. When signaled, the MCU extracts the packet from the FIFO and the boolean vector only if the CRC indicates a correct packet reception. One of the novelties of our architecture is the speed up of the rule matching part by the use of the boolean vector.

After receiving a signal from the hardware that a packet is ready, the MCU applies higher-level software packet filtering on the received packet. The MCU evaluates the expression tree produced from the filter specification. Some of the simple packet field matching rules are assigned to hardware (CL), and therefore, the outcomes of those rules already exist in the boolean vector. This contributes to the minimization of the delay for processing a packet and the power consumed by the MCU.

Clearly the MCU, being more flexible than the configurable logic, can perform more sophisticated operations on packets. Assisted by the CL, the MCU is capable of deciding if a packet is destined for the current node or if it must be forwarded, even for the cases that require complex calculations (e.g. distance calculation). Also the routing protocols that maintain little or no state on the sensor node are implemented in the MCU. Examples of these protocols are flooding without duplicate suppression.

The second tier of our architecture lies in the sensor node's central processing unit where the applications execute. A Router Manager is responsible for assigning the different sets of rules to the appropriate part (radio board, main CPU) and for delivering each packet to the appropriate application

All the routing protocols that need to maintain state for their proper functionality are implemented on the sensor node. In order to maintain this required state a routing agent should be executing on each node. The special functions that they need (e.g. CRC check, packet modification) can be provided by the first tier when the agents register with the Router Manager (Figure 2).

3. ANALYSIS AND MEASUREMENTS

In our prototype implementation an iKit2000[7] development board is connected to a WINS sensor node[11] through a serial port (115.2Kbps). In our case the built-in radio was not used. Instead our prototype radio board on the iKit2000 was used. The iKit2000 has a Triscend E520 [10] chip, which is essentially a 8032 microcontroller core as well as 40K of FPGA. The development board is connected to an RFM [8] radio module (10Kbps, 256-byte packets), which is the actual radio hardware used. The current stage of prototype implementation includes software packet transmission and reception as well as the first stage of packet processing in the 8032 MCU. The second stage of packet processing is performed on the CPU of the WINS node. We are also in the process of building the hardware filters in the FPGA part of the Triscend E520 device.

Next, we present an analytical study and measurement data for the prototype implementation. Our analytical study considers both the incurred delay and the energy consumed by one packet.

Suppose that D_{RX}, D_{TX} are the delays to receive and transmit a packet for the software receiver/transmitter respectively. Assume for the MCU that D_{MCUFW} is the filter processing delay for packets that are going to be forwarded to other nodes, D_{MCUAC} is the filter processing delay for packets that are actually accepted by a node, and D_{SR} is the delay of the serial port between the radio board and the main node processor. Also assume that the

87

corresponding delays for the main node CPU are D_{CPUFW} and D_{CPUAC} respectively. Pr_{AC} and Pr_{FW} are the corresponding probabilities of a packet being accepted and being forwarded respectively. We don't consider the case of the dropped packets because most dropped packets are duplicate packets that are detected in the main sensor node and will cross the serial link anyway.

The delays without and with filtering in the MCU are:

$$D_{NF} = (D_{RX} + D_{SR} + D_{CPUAC}) \cdot Pr_{AC} + (D_{RX} + D_{SR} + D_{CPUFW} + D_{SR} + D_{TX}) \cdot Pr_{FW}$$

$$D_F = (D_{RX} + D_{MCUAC} + D_{SR}) \cdot Pr_{AC} + (D_{RX} + D_{MCUFW} + D_{TX}) \cdot Pr_{FW}$$

The difference in delays is:

$$D_{diff} = D_{NF} - D_F = (D_{CPUAC} - D_{MCUAC}) \cdot Pr_{AC} + (2 \cdot D_{SR} + D_{CPUFW} - D_{MCUFW}) \cdot Pr_{FW}$$

Our prototype implementation measurements are shown in Table 2. Given this data and packet acceptance and forwarding probabilities Pr_{AC}, Pr_{FW} the difference in delays is:

$$D_{diff} = -2.0285 \cdot Pr_{AC} + 68.281 \cdot Pr_{FW}$$

In order for this difference to be zero Pr_{FW} must be $Pr_{FW} = 0.0297$ Pr_{AC} which is satisfied for the simulation data and for most practical sensor network protocols.

On the other hand the corresponding energy calculations are:

$$E_{NF} = (E_{RX} + E_{SR} + E_{CPUAC}) \cdot Pr_{AC} + (E_{RX} + E_{SR} + E_{CPUFW} + E_{SR} + E_{TX}) \cdot Pr_{FW}$$

$$E_F = (E_{RX} + E_{MCUAC} + E_{SR}) \cdot Pr_{AC} + (E_{RX} + E_{MCUFW} + E_{TX}) \cdot Pr_{FW}$$

And the difference in energy consumption is:

$$E_{diff} = E_{NF} - E_F = (E_{CPUAC} - E_{MCUAC}) \cdot Pr_{AC} + (2 \cdot E_{SR} + E_{CPUFW} - E_{MCUFW}) \cdot Pr_{FW}$$

If we assume that the power consumption of the main node CPU is α times the power consumption of the MCU and the serial port power consumption is the sum of the power consumption of the CPU and the MCU (because both devices should be on durinf serial port operation) then the difference in energy is:

$$E_{diff} = E_{NF} - E_F = (\alpha \cdot D_{CPUAC} - D_{MCUAC}) \cdot P_{MCU} \cdot Pr_{AC} + (2 \cdot (\alpha + 1) \cdot D_{SR} + \alpha \cdot D_{CPUFW} - D_{MCUFW}) \cdot P_{MCU} \cdot Pr_{FW}$$

Typical power consumption values for e.g. the Atmel AVR MCU and the E520 15mW and 470mw respectively. The actual power consumption of the WINS node is 351mW. This data result in two values of α: i) 23.4 for the WINS node/AVR combination and ii) 0.747 for the WINS node/E520 combination. For the two values of α the difference in energy is dominated by the D_{SR}, which is essentially the penalty paid for crossing the boundary between the radio board and the CPU.

The value of α is expected to be much greater than one in most cases of sensor nodes although in one (bad) case above α is less than one. This is due to the fact that the E520 device hosts a 8032 MCU and a small amount of FPGA on the same die. However, for

both values of α, the data on Table 1 and the probabilities of packets being accepted and forwarded (simulation data) we have values of E_{diff}, which are 1167 and 79 times the value of the MCU power consumption.

Table 1. Measured parameters on prototype

Parameter	Value(ms)
D_{MCUAC}	4.182
D_{MCUFW}	4.894
D_{CPUAC}	0.111
D_{CPUFW}	0.125
D_{SR}	36.532

4. CONCLUSIONS

As we have seen from the simulation data a considerable percentage of packets that enter a node are processed in a straightforward manner and are either redirected to the radio board, forwarded to the main processor or simply dropped. We proposed a two-tier architecture that enables the lower communication layers to perform the simple processing, drop or redirection of the packets as low as the radio board of a sensor node. As an additional feature, our architecture also enables the sensor applications to define methods for routing their own packets. We demonstrated a realization of the two-tier architecture in our prototype implementation, which includes a packet processor in the MCU of a system-on-a-chip.

5. ACKNOWLEDGEMENTS

6. REFERENCES

[1] Chien C. et al, "Design experience with an integrated testbed for wireless multimedia computing", MoMuC '96, p.231-8.

[2] Estrin D., Govindan R., Heidemann J., Kumar S., "Scalable Coordination in Sensor Networks", MobiCOM '99, pp. 263-70.

[3] Johnson D., Maltz D., "Dynamic Source Routing in Ad Hoc Wireless Networks", Mobile Computing, pp. 153-181, Kluwer Academic Publishers, 1996.

[4] McCanne S., Jacobson V., "The BSD Packet Filter: a New Architecture for User-level Packet Capture", Proceedings of the Winter 1993 USENIX Conference, pp. .259-69, 1993.

[5] Atmel FPSLIC, http://www.atmel.com/atmel/products/

[6] Cots Dust, http://www-bsac.eecs.berkeley.edu/~pister/SmartDust/

[7] Ikit2000 Development Kit, http://www.ikit2000.com

[8] RF Monolithics, http://www.rfm.com

[9] Sensoria Corporation, http://www.sensoria.com

[10] Triscend Corp, http://www.triscend.com

[11] WINS Architecture, http://wins.rsc.rockwell.com

Modulation Scaling for Energy Aware Communication Systems

Curt Schurgers
NESL, EE Dept., UCLA[†]
curts@ee.ucla.edu

Olivier Aberthorne
NESL, EE Dept., UCLA[†]
thornado007@yahoo.com

Mani B. Srivastava
NESL, EE Dept., UCLA[†]
mbs@ee.ucla.edu

ABSTRACT

In systems that require low energy consumption, voltage scaling is an invaluable circuit technique. It also offers energy awareness, trading off energy and performance. In wireless handheld devices, the communication portion of the system is a major power hog. We introduce a new technique, called modulation scaling, which exhibits benefits similar to those of voltage scaling. It allows us to trade off energy against transmission delay and as such introduces the notion of energy awareness in communications. Throughout our discussion, we emphasize the analogy with voltage scaling. As an example application, we present an energy aware wireless packet scheduling system.

Keywords

energy awareness, adaptive modulation, scaling

1. INTRODUCTION

In tetherless battery-operated devices, power consumption is a critical design aspect. It has been realized that it is **energy awareness**, in addition to low power, that is required for most applications [1]. Scaling the supply voltage is the most common circuit technique to offer both low energy consumption and energy awareness [2]. In operating system research, the clock speed and supply voltage are dynamically adjusted based on the predicted workload [3]. Another approach, proposed for self-timed [4] and synchronous [5] systems, is to use the amount of buffered load to steer the adaptation.

Furthermore, a lot of these battery-operated devices are equipped with a wireless communication subsystem. A major source of their energy consumption is the actual data transmission over the air. Despite the work on energy awareness in digital electronic circuits, it has been overlooked that the same tradeoffs are present in communications as well. In this paper, we show that the **modulation can be scaled** much the same way as operating voltage can, reducing the overall energy consumption for transmitting each bit. Although the basic idea of changing the modulation on the fly has been used to increase the throughput in the presence of fading channels [6], it has never been exploited for low power purposes. We have applied this principle towards an **energy aware wireless scheduling system**.

2. COMMUNICATION THEORY

Since we investigate the relationship between modulation and transmission speed, we first need to derive the relevant expressions. We focus on Quadrature Ampitude Modulation (QAM) due to its ease of implementation and analysis [6]. However, our techniques are perfectly extendable to other modulation schemes, only the formulas and curves will change accordingly. The performance of QAM in terms of Bit Error Rate (BER) is given by (1)-(3) [7].

$$BER = \frac{4}{b} \bullet \left(1 - \frac{1}{2^{b/2}}\right) \bullet Q\left(\sqrt{3 \bullet \frac{SNR}{2^b - 1}}\right) \tag{1}$$

$$SNR = \frac{P_S}{P_n} \bullet A \tag{2}$$

$$P_n = N_0 \bullet \psi \bullet R_S \tag{3}$$

The constellation size in number of bits per symbol is represented by b. The received Signal to Noise Ratio (*SNR*) is defined as (2), where P_S is the transmit power and A contains all transmission loss components. The noise power P_n is a function of the symbol rate R_s, the noise power spectral density N_0 and a factor ψ that takes into account all other elements, such as filter non-idealities. [7]. We can manipulate these equations to obtain the following expression for the required transmit power:

$$P_S = C_S \bullet R_S \bullet \left(2^b - 1\right) \tag{4}$$

$$C_S = \frac{N_0 \bullet \psi}{A} \bullet \Gamma \tag{5}$$

$$\Gamma = \left(\frac{1}{3}\right) \bullet \left[Q^{-1}\left(\left(\frac{1}{4}\right) \bullet \left(1 - \frac{1}{2^{b/2}}\right)^{-1} \bullet b \bullet BER\right)\right]^2 \tag{6}$$

Since our goal is to investigate the energy-delay characteristics while varying the communication parameters we want to keep the system performance constant for a fair comparison. In practical scenarios it makes sense to operate at a target BER. Due to the inverse $Q(.)$ function in (6), C_s is only a weak function of b.

An energy aware communication system **adjusts b and R_s** to reduce the overall energy. The transmit power P_S (delivered mainly by the power amplifier), however, is not the only source of

power spending. Electronic circuitry for filtering, modulating, upconverting, etc. contributes as well. Equation (7) expresses this component P_E for a system that can dynamically change the symbol rate [8]. Parts of the circuitry operate at a frequency that follows the instantaneous symbol rate, while other parts have a fixed frequency proportional to the maximum symbol rate. The proportionality factors and switching activity are all incorporated in C_A and C_B.

$$P_E = \left[C_E + C_R \bullet \frac{R_{S_{max}}}{R_S} \right] \bullet R_S \tag{7}$$

$$C_E = C_A \bullet V^2 \qquad C_R = C_B \bullet V^2 \tag{8}$$

The total power consumption is the sum of both the transmit and electronics power. As in digital circuit design, it makes more sense to look at the energy consumption rather than the total power. We can express the energy to transmit one bit, E_{bit}, as:

$$E_{bit} = (P_S + P_E) \bullet T_{bit} \tag{9}$$

In this equation, T_{bit} is the time it takes to transmit one bit. The goal is to minimize the energy per bit by choosing the correct values of b and R_s. For typical applications, however, we need to constrain the total delay a packet may incur, translating to a bound on T_{bit}. The optimization problem can be summarized as:

$$\min \quad E_{bit} = \left[C_S \bullet (2^b - 1) + C_E + C_R \bullet \frac{R_{S_{max}}}{R_S} \right] \bullet \frac{1}{b} \tag{10}$$

$$T_{bit} = \frac{1}{b \bullet R_S} = T_{max} \tag{11}$$

3. PERFORMANCE TRADEOFFS

Our numerical results in this section are based on table 1. The values of C_S, C_E and C_R are extracted from [8], which describes the actual implementation of an adaptive QAM system. Figure 1 depicts E_{bit} as a function of b and R_s as obtained from (10). The corresponding values of T_{bit} from (11) are shown in figure 2. Based on these two figures, we can evaluate the performance in terms of energy consumption for varying constraints on the delay (i.e. varying T_{max}).
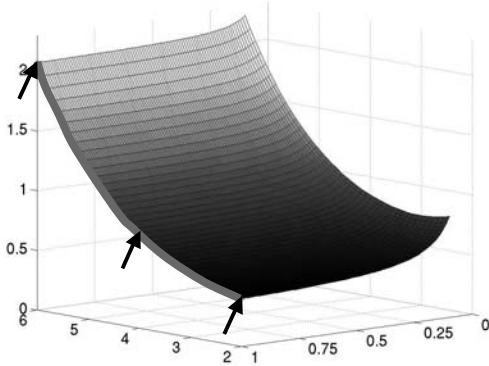
Figure 1: Energy consumption for adaptive R_s system

R_{Smax}	1 MHz	C_S (b=4)	10^{-7}
BER	10^{-5}	C_E	$8. \, 10^{-8}$
		C_R	10^{-7}
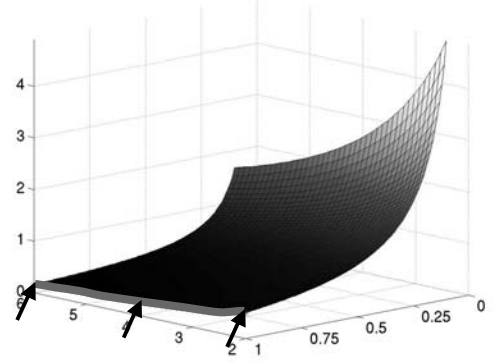
Figure 2: Delay per bit

From these figures, it is clear that operating at the maximum R_S is preferable for any b. This is logical as this results in both a lower T_{bit} and a lower E_{bit}. The symbol rate should therefore be chosen as high as possible, considering implementation issues and their power penalties. Varying the constellation size b is the only option to trade off energy versus delay. In practice, b does not have an infinitesimal granularity but typically only takes on even integers, indicated by the black arrows in figures 1 and 2.

Note that the results of figure 1 are for a communication system that has provisions to vary the symbol rate on the fly. In (7), this introduces the term with constant C_R. Since the optimal symbol rate is always the maximum one, a variable symbol rate provision is not needed for energy awareness reasons. In fact, the **system can be designed for a fixed symbol rate** instead. The circuitry that is described by the term with constant C_R is still present of course. We therefore cannot simply remove this term. However, we modify equation (10) by setting R_{Smax} equal to R_S, such that the energy per bit is now expressed as:

$$\min \quad E_{bit} = \left[C_S \bullet (2^b - 1) + C_E + C_R \right] \bullet \frac{1}{b} \tag{12}$$

Upon investigating (12), it is clear that E_{bit} is no longer a function of the symbol rate. Since a higher R_s still results in a lower T_{bit}, it is still beneficial to operate at the highest symbol rate that can be implemented efficiently. The reason is that besides the advantage of lower delays, this would also improve the capacity if the wireless medium were shared. We can visualize the energy and delay curves by taking the intersection of the surface in figures 1 and 2 with a plane at $R_S = 1$ MHz.

It is clear that energy and delay can be traded off against each other by varying b. In analogy with voltage scaling techniques in digital circuits, we refer to this process as **modulation scaling**. Depending on the delay that is acceptable, the constellation size can be adapted to meet that constraint with the minimum amount of energy. If this adaptation is performed on the fly, it results in **energy awareness**.

4. COMPARISON BETWEEN VOLTAGE SCALING AND MODULATION SCALING

The equations in the previous sections resemble those of voltage scaling, yet there are some key differences. It is important to highlight these differences, as they also contribute to a physical understanding of the tradeoffs of modulation scaling. Figure 3 places both scaling techniques next to each other. In the equations for voltage scaling, P_S is the switching power and P_L the leakage power [3]. It is clear that **the functionality of supply voltage V corresponds to that of the constellation size b** (hence the terms voltage and modulation scaling). In the left column, the energy is only dependent on b and not on R_B. Equivalently in the right column, the energy term due to the switching power (P_S/f) depends on V and not on f. There is however a crucial difference, regarding the interpretation of time.

In the digital circuit case, the total effective delay for an operation t_d has to be smaller than $1/f$. Similarly in a communication system the total time it takes to transmit a packet (or a bit) has to be smaller than a certain maximum value. **The difference between both systems, however, is the period over which energy is consumed**. In a communication system, the power has to be multiplied by the **effective time** of the operation. In digital circuits, on the other hand, the power is multiplied by the cycle time, which is in effect the maximum delay. As such, there is no true one-to-one mapping between R_B (or R_S for that matter) and f. However, when considering R_B and f as constants of the system, they result in a lower bound on b or V in similar ways (see the third line of equations in figure 3).

5. ENERGY AWARE WIRELESS PACKET SCHEDULING

Like energy aware OS scheduling, we can perform energy aware packet scheduling. We study the communication system setup depicted in figure 4, which consists of a point-to-point transmission link. Packets arrive at the sender and possibly need to be buffered before transmission. We assume that both the packet sizes and the intervals between packet arrivals, called inter-arrival times, follow an exponential distribution. Without modulation scaling, this setup corresponds to the well-known M/M/1 queuing system [9].
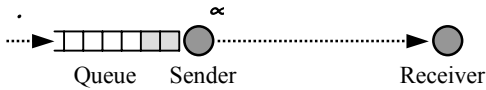
Figure 4: Setup of the queuing system

The average packet arrival rate is denoted by $.$. The inverse of the average service time is called the service rate, $\propto$, which gives the average number of packets that can be sent per unit time. It is expressed by (13), where L is the average packet size.

$$\propto = \frac{R_B}{L} = \frac{b \bullet R_S}{L} \tag{13}$$

Because of the statistical properties of inter-arrival and service times, the number of packets in the buffer may vary considerably. Most of the time, the buffer is empty. In those situations, it is beneficial to scale the modulation down to conserve energy. When the buffer starts to fill up, we can increase b to avoid long queuing times or buffer overflow. This kind of system therefore is a good candidate for modulation scaling. A similar observation has been made for digital circuits, where a queue is introduced to average the rate over several samples in a DSP system [5].

The idea is to choose the constellation size based on the number of packets in the system (i.e. being transmitted or in the queue), which we refer to as the system state. For each state $S_n = n$, we have a particular constellation size b_n, which translates into a value of $\propto_n$ through equation (13). The collection of $\{b_n\}$ for all the possible states determines the average energy consumption and delay of the queuing system. Our goal is to find which $\{b_n\}$ minimizes the energy for a particular delay constraint. We can analyze this problem using queuing theory. In steady state, the probability of being in state n can be expressed as [9]:

$$P_n = P_0 \bullet \frac{\dot{}^{\,n}}{\displaystyle\prod_{k=1} \propto_k} \tag{14}$$

In this equation, P_0 is a constant such that the sum of P_n over all states is equal to 1. We assume an infinite buffer size, which is a reasonable approximation for real systems, as memory has become rather inexpensive for these applications. For each state the energy consumption per bit is given by (12).

The average energy per packet, E_{av}, is the ratio of the average power per packet and the packet arrival rate. The average power is the product of the probability P_n of being in a state, the rate $\propto_n$ in that state and the average energy per packet ($E_n.L$) in that state:

$$E_{av} = \frac{P_{av}}{\dot{}} = \frac{1}{\dot{}} \bullet \sum_{n=1}^{8} P_n \bullet \propto_n \bullet E_n \bullet L \tag{15}$$

Since (13) holds in every state, we can simplify this expression to:

Communications	Digital Circuits
$P_S = C_S \bullet G_S \bullet R_B \bullet \dfrac{(2^b - 1)}{b}$	$P_S = \alpha \bullet C_L \bullet f \bullet V^2$
$P_E = [C_E + C_R] \bullet \dfrac{R_B}{b}$	$P_L = V \bullet I_0 \bullet e^{\frac{V}{nV_T}}$
$R_B = \dfrac{1}{T_{max}} \Rightarrow b = \dfrac{1}{R_S \bullet T_{max}}$	$\dfrac{1}{t_d} = f \Rightarrow \dfrac{(V - V_T)^2}{V} = \dfrac{C_L}{k} \bullet f$
$E = (P_S + P_E) \bullet \dfrac{1}{R_B}$	$E = (P_S + P_L) \bullet \dfrac{1}{f}$

Figure 3: Comparison between adaptive modulation and voltage scaling

$$E_{av} = \frac{R_S}{\cdot} \bullet \sum_{n=1}^{8} P_n \bullet E_n \bullet b_n \qquad (16)$$

Furthermore, queuing theory tells us that we can express the average delay of a packet as [9]:

$$T_{av} = \frac{1}{\cdot} \bullet \sum_{n=0}^{8} n \bullet P_n \qquad (17)$$

For our numerical evaluation and subsequent simulations, we have again chosen the settings of table 1, augmented with those of table 2. In figure 5 each point on the energy versus delay curve represents the average performance of the queuing system for a particular set of $\{b_n\}$. We have only plotted those operating points that minimize the energy for a delay constraint. The dashed curve is for the ideal system that would allow fractional values of b. In practical situations, we select b from the set of even integers, which results in the curve labeled 'Queuing'. Table 3 gives the values of $\{b_n\}$ for the operating point indicated by the arrow.

However, even this system is difficult to implement in practice, because the modulation would have to be adjusted every time the system state changes. This means that the constellation size can change when either a packet enters or leaves the queuing system. On the other hand, the receiver needs to know what modulation scheme the sender is using in order to decode the symbols and every change in constellation size needs to be communicated to the receiver.

In practice, it is more appropriate to **adapt the constellation size only at the beginning of the packet transmission**. An indicator (encoded with a fixed modulation) in the packet header that describes the modulation used for the packet payload. The modulation scaling is performed based on the number of packets in the queue at the time the transmission starts. The curve labeled 'Simulation' in figure 5 presents the performance of such a practical scheme (it includes the overhead due to the indicator). There is a penalty compared to the theoretical queuing system since the modulation is only adapted when a packet starts being transmitted, instead of every time the number of packets in the system changes. For this practical system, we can select the best operating point for each delay constraint from the curve in figure 5. This operating point defines the values of $\{b_n\}$ that have to be chosen.
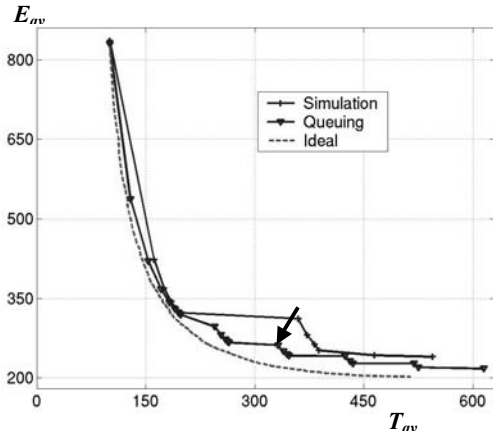
Figure 5: Energy-delay tradeoff for an energy aware queuing system

Table 2: Simulation settings

. (packets/s)	5000
L (bits)	400
$\propto_n$ (packets/s)	$2500 \cdot b_n$

Table 3: Settings for an example operating point

S_n	1	2	3	4	5	≥ 6
b_n	2	4	4	4	6	6

6. CONCLUSIONS

We have presented modulation scaling, which allows us to design energy aware communication systems. We have highlighted the similarities and differences compared to voltage scaling used in digital circuits. A lot of approaches that have been explored in the context of voltage scaling can be applied to modulation scaling as well. We have investigated this for energy aware wireless packet scheduling. However, many other applications can be envisioned that benefit from modulation scaling. Also techniques that improve the system's energy performance can be incorporated into this framework, such as parallelism.

7. ACKNOWLEDGMENTS

8. REFERENCES
[1] Graybill, R., "DARPA Power Aware Computing/ Communication," *http://www.darpa.mil/ito/research/pacc/.*

[2] Chandrakasan, A., Sheng, S., Brodersen, R., "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, Vol.27, pp. 473-484, Dec..

[3] Govil, K., Chan, E., Wasserman, H., "Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU," *MobiCom'95*, Berkeley, CA, pp. 13-25, Nov. 1995.

[4] Nielsen, L., Niessen, C., Sparsø, J., van Berkel, K., "Low Power Operation Using Self-Timed Circuits and Adaptive Scaling of the Supply Voltage," *Trans. on VLSI Systems*, Vol.2, No.4, pp. 391-397, Dec. 1994.

[5] Gutnik, V., Chandrakasan, A., "Embedded Power Supply for Low-Power DSP," *Trans on VLSI Systems*, Vol.5, No.4, pp. 425-435, Dec. 1997.

[6] Ue, T., Sampei, S., Morinaga, N., Hamaguchi, K., "Symbol Rate and Modulation Level-Controlled Adaptive Modulation/TDMA/TDD System for High-Bit Rate Wireless Data Transmission," *Trans. on Vehicular Technology*, Vol.47, No.4, pp. 1134-1147, Nov. 1998.

[7] Proakis, J., "Digital Communications," *McGraw-Hill Series in Electrical and Computer Engineering*, 3rd Edition, 1995.

[8] Cho, K., Samueli, H., "A 8.75-MBaud Single-Chip Digital QAM Modulator with Frequency-Agility and Beamforming Diversity," *Proc. of the IEEE 2000 Custom Integrated Circuits Conference*, Orlando, FL, pp. 27-30, May 2000.

[9] Bertsekas, D., Gallager, R., "Data Networks," *Prentice Hall*, 2nd Edition, 1999.

ENERGY EFFICIENT ROUTING IN WIRELESS SENSOR NETWORKS

Curt Schurgers
Mani B. Srivastava

Networked & Embedded Systems Lab (NESL), Electrical Engineering Department
University of California at Los Angeles (UCLA), CA

ABSTRACT

Wireless sensor nodes can be deployed on a battlefield and organize themselves in a large-scale ad-hoc network. Traditional routing protocols do not take into account that a node contains only a limited energy supply. Optimal routing tries to maximize the duration over which the sensing task can be performed, but requires future knowledge. As this is unrealistic, we derive a practical guideline based on the energy histogram and develop a spectrum of new techniques to enhance the routing in sensor networks. Our first approach aggregates packet streams in a robust way, resulting in energy reductions of a factor 2 to 3. Second, we argue that a more uniform resource utilization can be obtained by shaping the traffic flow. Several techniques, which rely only on localized metrics are proposed and evaluated. We show that they can increase the network lifetime up to an extra 90% beyond the gains of our first approach.

I. INTRODUCTION

Recently IC and MEMS have matured to the point where they enable the integration of communications, sensors and signal processing all together in one low-cost package. It is now feasible to fabricate ultra-small sensor nodes that can be scattered on the battlefield to gather strategic information [1]. The events detected by these nodes need to communicated to gateways or users who tap into the network. This communication occurs via multi-hop routes through other sensor nodes. Since the nodes need to be unobtrusive, they have a small form-factor and therefore can carry only a small battery. As a result, they have a limited energy supply and low-power operation is a must. Multi-hop routing protocols for these networks necessarily have to be designed with a focus on energy efficiency.

The proposed approaches lean towards localized algorithms [1][2]. Due to the large number of sensors, network-scale interaction is indeed too energy expensive. Moreover, a centralized algorithm would result in a single point of failure, which is unacceptable in the battlefield. In this paper, we propose two options for localized algorithms to increase the sensor network lifetime: (1) minimize the energy consumption of transmissions and (2) exploit the multi-hop aspect of network communications.

The first option is to combine/fuse data generated by different sensors [1][2]. In [3] cluster head selection is proposed to perform this task. However, in section IV, we present a robust way of achieving the same functionality without explicit cluster formation.

The second option focuses on the paths that are followed during the data routing phase. The framework presented in [2] advocates a localized model called 'directed diffusion'. Other work uses information on battery reserve and the energy cost to find the optimal routes [4]. The routing protocol in [5] is based on the node's location, transmit energy and the residual battery capacity. In contract to this prior work, we propose a guideline that aims at spreading the network traffic in a uniform fashion. Our spreading ideas, although partly tailored towards the underlying routing algorithm we have chosen, should be beneficial for the energy aware routing protocols mentioned above. We discuss these spreading techniques in section V.

However, before discussing our data fusion and spreading, we first focus on the problem statement: how to increase the lifetime of a network of energy constrained devices. This results in a practical guideline, which considers the energy histogram. All of this is treated in section II.

II. PROBLEM STATEMENT

1. Energy Optimal Routing

Traditional ad-hoc routing algorithms focus on avoiding congestion or maintaining connectivity when faced with mobility [6]. They do not consider the limited energy supply of the network devices. The example of figure 1 illustrates how the limited supply alters the routing issue. Nodes A and E first send 50 packets to B. Afterwards, F sends 100 packets to B. From a load balancing perspective, the preferred paths are ADB, ECB and FDB respectively.

However, when the nodes are energy constrained such that they can only send 100 packets, these paths are no longer optimal. Indeed, D would have used up 50% of its energy

before it can forward packets from *F* to *B*. In this case, all packets could have been delivered by choosing paths *ACB*, *ECB* and *FDB*. If, instead of *F*, node *C* would have become active, *A* should have used the original path *ADB*.
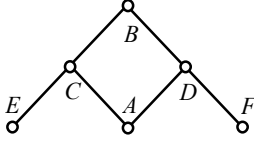
Figure 1: Load versus energy oriented routing

This simple case study highlights the following crucial observation: **optimal traffic scheduling** in energy constrained networks requires future knowledge. In our example, a maximum number of packets can reach *B* only if right from the start we know exactly when (and which) nodes will generate traffic in the future.

2. Energy Efficient Routing

Ideally, we would like the sensor network to perform its functionality as long as possible. Optimal routing in energy constrained networks is not practically feasible (because it requires future knowledge). However, we can soften our requirements towards a statistically optimal scheme, which maximizes the network functionality considered over all possible future activity. A scheme is **energy efficient** (in contrast to 'energy optimal') when it is statistically optimal and causal (i.e. takes only past and present into account).

In most practical surveillance or monitoring applications, we do not want any coverage gaps to develop. We therefore define the **lifetime** we want to maximize as the worst-case time until a node breaks down, instead of the average time over all scenarios. However, taking into account all possible future scenarios is too computationally intensive, even for simulations. It is therefore certainly unworkable as a guideline to base practical schemes on. Considering only one future scenario leads to skewed results, as shown in the example of figure 1.

3. Traffic Spreading Rationale

To derive a practical guideline, we start from the following observation: the minimum hop paths to a user for different streams tend to have a large number of hops in common [7]. Nodes on those paths die sooner and therefore limit the lifetime of the network. Figure 2 presents a typical energy consumption histogram at a certain point in time. Some nodes have hardly been used, while others have almost completely drained their energy.
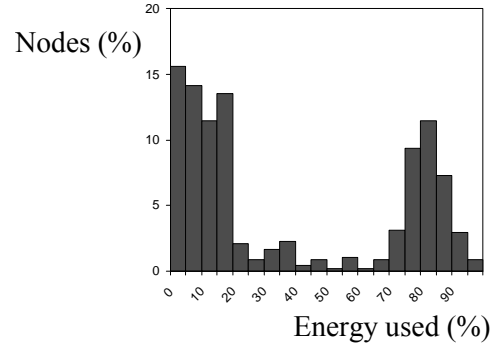
Figure 2: Undesirable energy histogam

As nodes that are running low on energy are more susceptible to die sooner, they have become more critical. If we assume that all the nodes are equally important (we revisit this assumption in section V.2), no node should be more critical than any other one. At each moment every node should therefore have used about the same amount of energy, which should also be minimized. The histogram of figure 3 is thus more desirable than the one of figure 2, although the total energy consumption is the same.
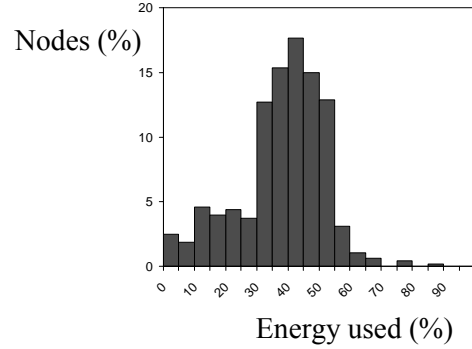
Figure 3: Desirable energy histogram

Striving for a compact energy histogram translates into the guideline that traffic should be spread over the network as uniformly as possible. Since visualizing the histogram over time is hard, we propose to use the root mean square E_{RMS} as an indicator instead (the lower this value, the better). It provides information on both the total energy consumption and on the spread.

III.   BASIC ROUTING

As an underlying routing scheme, we base ourselves on the paradigm of directed diffusion [2]. When a user taps into the sensor network, he announces the type of information he is interested in. While flooding this 'interest' possibly using techniques like SPIN [8], gradients are established in each node. These gradients indicate the 'goodness' of the

different possible next hops and are used to forward sensor data to the user.

We have opted for a simple instantiation of this paradigm, which we call Gradient-Based Routing (GBR). While being flooded, the 'interest' message records the number of hops taken. This allows a node to discover the minimum number of hops to the user, called the node's **height**. The difference between a node's height and that of its neighbor is considered the gradient on that link. A packet is forwarded on the link with the largest gradient. Although our techniques to increase the network lifetime are built upon GBR, the main principles are general enough to also be applicable to other ad-hoc routing protocols.

IV.   DATA COMBINING

1. Data Combining Entities (DCE)

Individual sensor nodes process their sensor data before relaying it to the user [1]. It is advantageous to combine observations from different nodes to increase the resource efficiency. This process reduces not only the header overhead, but also the data itself can be compacted as it contains partly the same information.

Although this combining can be implemented by explicitly selecting a cluster head [3], we present a scheme that is more robust to random node failures. First note that sensor nodes that are triggered by the same event, are typically located in the same vicinity. The resulting cloud of activated nodes is also in close communication proximity. The routes from these nodes to the user merge early on [7]. Nodes that have multiple streams flowing through them can create a Data Combining Entity (DCE), which takes care of the data compaction. Simulations have shown that the DCEs are located inside or very close to this cloud of activated nodes.

This scheme is highly robust. When a node with a DCE dies, the packets automatically take an alternative route and pass through another node that can create a new DCE.

2. Simulations

Figure 4 depicts the effects of our DCE-based data compaction on the total energy consumption. The nodes in this simulation are distributed randomly over a rectangular area with a constant width of 32 m and a linearly increasing length B. The radio transmission range R is 20 m and the average node density is kept constant at $10^{-2}/m^2$. The nodes at the top of this area sense a target and notify a user that is located at the bottom end (the transmission of one packet takes 5.76 $\propto$J). For our numerical results, we assume that a packet that is combined with another one can

be compressed to 60% of its original size. We consider 3 distinct cases: without DCE, with at most one DCE (a compression bit in the packet header signals if the packet has been compressed already) on each route to the user and with no restrictions on the number of DCEs. The reduction in energy consumption is as expected (up to a factor 2 to 3), linearly proportional to the number of bits sent.
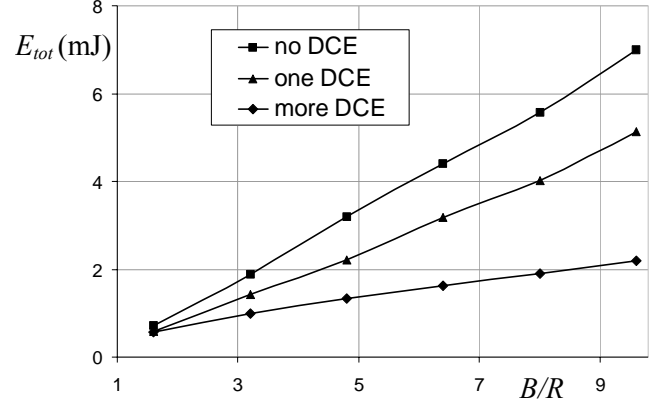
Figure 4: Energy comparison for DCE
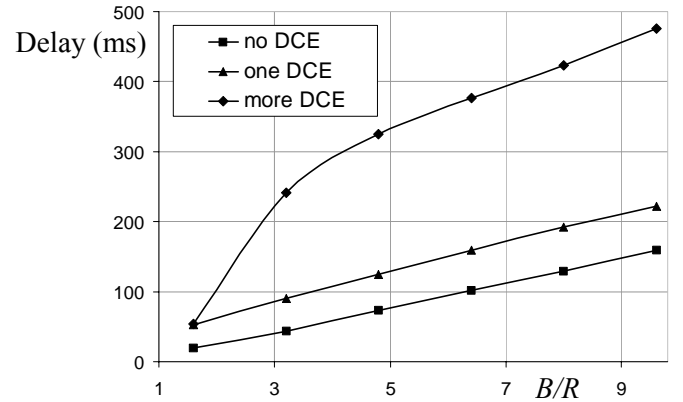
Figure 5: Delay comparison for DCE

The flip side is the average delay per packet, which is presented in figure 5. Since DCEs have to buffer data for a while, the packet delay will increase with the number of combining stages applied. Whether or not this is acceptable depends on the application.

V.   NETWORK TRAFFIC SPREADING

1. Spreading Techniques

Stochastic Scheme: Using a rationale similar to the one of [9], each node can select the next hop in a stochastic fashion. More specifically, when there are two or more next hops with the same lowest gradient, a random one is

chosen. This does not increase the length of the path followed, but nonetheless contributes to spreading the network traffic.

Energy-based Scheme: When a node detects that its energy reserve has dropped below a certain threshold (50% in our simulations), it discourages others from sending data to it by increasing its height. This may change a neighbor's height (since a node's height is one more than that of its lowest neighbor). It in turn informs other nodes and these updates are propagated as far as is needed to keep all the gradients consistent.

Stream-based Scheme: The idea is to divert new streams away from nodes that are currently part of the path of other streams. A node that receives packets tells all its neighbors except to the one from where the stream originates, that its height has increased. Again, other nodes must make sure the gradients remain consistent. As a result of this scheme, the original stream is unaffected, since those nodes have not updated the height of the next hop. New streams of packets, however, will take other paths as the height of the nodes on the first path has apparently increased.

2. Simulations
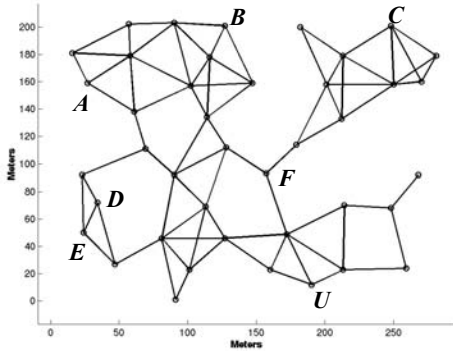
Figure 6: Wireless sensor network topology

Scenario 1: Nodes A and B (see figure 6) detect a different target and send packets to the user at regular intervals. After generating 100 packets each (this takes 11.8 seconds), these targets disappear and both nodes become inactive again. At this time, no node has been drained yet completely and the network connectivity is still fully intact. We have assumed a node has only 0.76 mJ of energy at its disposal (which is enough to send about 140 packets). The results can readily be scaled towards more realistic scenarios. Figure 7 shows the evolution of E_{RMS} as a function of time, for 5 different schemes. The unenhanced GBR is called 'standard'. Besides the three schemes discussed in V.1, we have also studied a combination of the stochastic and energy-based one.
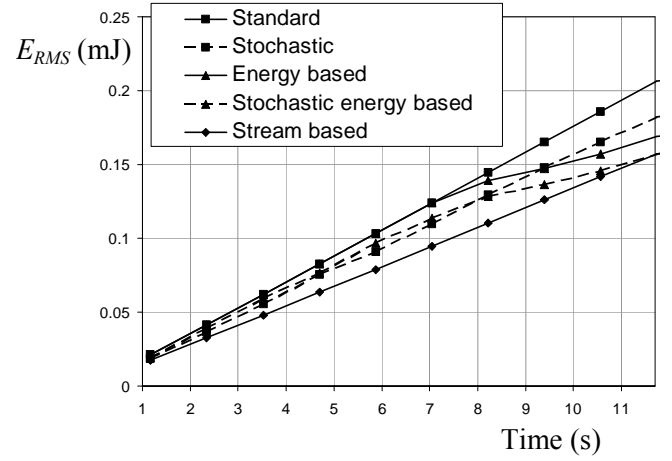
Figure 7: E_{RMS} for scenario 1

It is clear that the stream-based scheme indeed spreads the traffic more uniformly over the network. As soon as the energy of some nodes drops below 50%, the energy-based scheme kicks in. The stochastic routing provides an improvement both on top of the normal GBR and on top of the energy-based scheme.
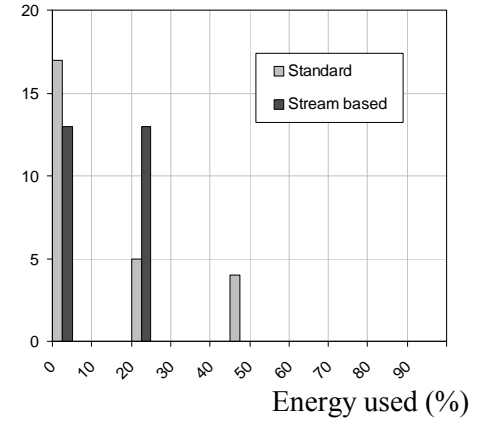
Figure 8: Energy histogram after 6 seconds

To verify that the E_{RMS} captures the relevant information, figure 8 shows the energy histogram for the standard and the stream-based scheme after 7 seconds. It is clear that spreading balances the energy consumption better.

Finally, we would like to show that the improved energy histogram is able to extend the network lifetime for a particular future scenario, although this does not prove anything about other possible futures. After 11.8 seconds node C starts forwarding packets to the user. Table 1 shows that the schemes that resulted in better traffic spreading also increase total traffic that reaches the user. We have verified that the time the network remains intact is increased by 90% when using the stream-based scheme.

Scheme	Packets received
Standard	127
Stochastic	133
Energy-based	160
Stochastic energy-based	161
Stream-based	175

Table 1: Packets received for scenario 1

Scenario 2: Nodes *D* and *E* (figure 6) each send 100 packets to the user in 11.8 s. Figure 9 illustrates that our traffic spreading schemes again result in a more uniform utilization of the network resources.
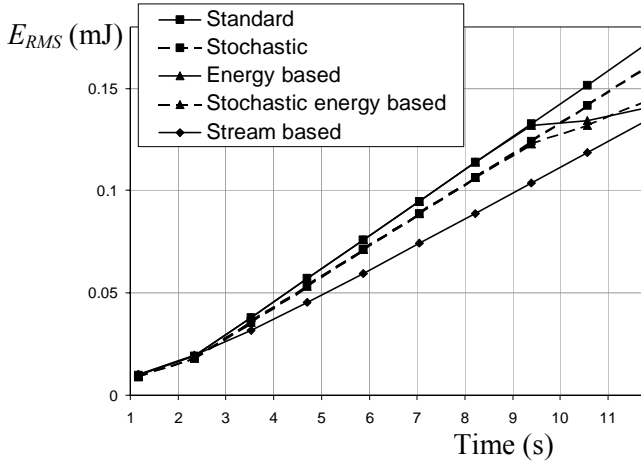
Figure 9: E_{RMS} for scenario 2

As before, we investigate one particular future activity scenario: node *C* becomes active after 11.8 seconds. From table 2, we conclude that spreading the network traffic has a negative effect as fewer packets are received! This is because the route taken by the standard GBR protocol avoids bottleneck node *F*. On the other hand, preading the traffic of *D* and *E* diverts some packets via *F* and therefore already partly drain this node before *C* can use it. This illustrates that spreading might increase the lifetime, although this does improve all possible futures. We observe however that the problems in this case are largely due to the fact that node *F* is critical as it is the only gateway to an entire subnet. Enhanced spreading techniques should therefore try to avoid critical nodes.

Scheme	Packets received
Standard	217
Stochastic	211
Energy-based	193
Stochastic energy-based	193
Stream-based	176

Table 2: Packets received for scenario 2

VI. CONCLUSIONS

In this paper we have argued that optimal routing in sensor networks is infeasible. We have proposed a practical guideline that advocates a uniform resource utilization, which can be visualized by the energy histogram. We acknowledge however that this is only a first cut at tackling this complicated issue. For example, exceptions must be made when nodes are critical in the overall network connectivity. We also propose a number of practical algorithms that are inspired by this concept. Our DCE combining scheme reduces the overall energy, while our spreading approaches aim at distributing the traffic in a more balanced way. We note that although we have started from GBR, our basic ideas and techniques should be able to enhance other routing protocols as well.

REFERENCES

[1] Sohrabi, K., Gao, J., Ailawadhi, V., Pottie, G., "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Communications Mag.*, Vol.7, No.5, pp.16-27, Oct. 2000.

[2] Estrin, D., Govindan, R., "Next Century Challenges: Scalable Coordination in Sensor Networks," *MobiCom'99*, Seattle, WA, pp.263-270, Aug. 1999.

[3] Rabiner, W., Chandrakasan, A., Balakrishnan, H., "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Hawaii International Conference on System Sciences*, Maui, HI, pp.10-19, Jan. 2000.

[4] Chang, J.-H., Tassiulas, L., "Energy Conserving Routing in Wireless Ad-Hoc Networks.," *INFOCOM'00*, Tel Aviv, Israel, pp.22-31, Mar. 2000.

[5] Stojmenovic, I., Lin, X., "Power-Aware Localized Routing in Wireless Networks," *Proceedings IPDPS 2000*, Cancun, Mexico, pp. 371-376, May 2000.

[6] Broch, J., Maltz, D., Johnson, D., Hu, Y. Jetcheva, J., "A performance comparison of multi-hop wireless ad-hoc network routing protocols," *Mobicom'98*, Dallas, Texas, pp.85-97, Oct. 1998.

[7] Pearlman, M., Haas, Z., "Improving the Performance of Query-Based Routing Protocols through 'Diversity-Injection'," *WCNC'99*, New Orleans, LA, pp. 1546-1550, Sept. 1999.

[8] Rabiner, W., Kulik, J., Balakrishnan, H., "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *MobiCom'99*, Seattle, WA, pp. 174-185, Aug. 1999.

[9] Dattatreya, G.R., Kulkarni, S.S., "Simulation of Adaptive Statistically Multiplexed Routing in Ad Hoc Networks," *WCNC'99*, New Orleans, LA, pp. 931-935, Sept. 1999.

Modulation Scaling for Real-Time Energy Aware Packet Scheduling

Curt Schurgers   Vijay Raghunathan   Mani B. Srivastava

Networked and Embedded Systems Lab (NESL), EE-Dept., University of California at Los Angeles (UCLA)
{curts,vijay,mbs}@ee.ucla.edu

Abstract— Portable wireless communication systems operate on a limited battery supply, and energy efficiency is therefore crucial. Voltage scaling techniques have been proposed to lower the energy consumption of embedded processors and real-time operating systems have incorporated these schemes in their task scheduling engine. However, the actual data transmission itself constitutes a major portion of the total energy consumption in these wireless communication systems. In this paper, we extend the scaling notion to the realm of wireless communications and propose a novel technique called modulation scaling to decrease the energy consumed during data transmission. Modulation scaling trades off energy consumption against transmission delay and as such, introduces the concept of energy awareness in communications. We investigate how modulation scaling can be exploited to design a dynamic power management engine at the level of the radio. This engine coordinates the packet transmission schedule while optimizing energy efficiency. We demonstrate such a power management module for real-time traffic and show that it reduces the energy consumption of data transmissions by up to 50% through smart traffic scheduling.

I.   INTRODUCTION

A.   Energy Awareness

Due to the proliferation of tetherless battery-operated devices, power consumption has become a critical aspect in system design. Recently people have come to realize that it is **energy awareness**, in addition to low power, that is required for most applications [1]. Systems should not be designed based on worst case operating conditions alone, but should dynamically adjust their performance to meet the required level, resulting in energy savings. Dynamically scaling the supply voltage is the most popular digital circuit technique to offer both low energy consumption and energy awareness [2].

Another trend is that wireless communication handhelds have replaced computers as the frontrunners of technology innovations. In addition, networks of embedded autonomous devices (called sensor networks) are on the verge of providing ubiquitous access and sampling to our environment [3]. In all these wireless battery operated systems, a major source of energy consumption is the actual data transmission over the air. Despite the work on energy awareness in digital electronic circuits, it has been overlooked that the same tradeoffs are present in communications as well. We show that the **modulation can be scaled** much the same way as operating voltage can, reducing the overall energy consumption for transmitting each bit. Although the basic idea of changing the modulation on the fly has been used to

increase the throughput in the presence of fading channels [4], it has never been exploited for low power purposes.

B.   Energy Aware Scheduling

How scaling brings about energy awareness depends on its integration into **dynamic power management**. Power management algorithms orchestrate the operation of the different system components to reduce energy consumption while meeting a specified minimum performance level. One frequently used power management technique is shutting down unused system components. However, it has been realized that dynamically adjusting the clock speed and operating voltage of the processor offers higher energy savings. Embedded processors frequently operate on real-time traffic such as multimedia streams and hence, the power management policy has to take the associated timing constraints (i.e. task deadlines) into account. Dynamic voltage scaling for real-time operating systems (RTOS) in the presence of deadlines has been studied in [5][6].

As the embedded processor is not the only critical component in wireless communication devices, we need to extend the **dynamic power management to the communication sub-system** as well. Indeed, the processed data streams have to be scheduled for transmission, typically under real-time constraints. This is true for multimedia handhelds and embedded monitoring devices such as sensor networks. We extend the ideas of dynamic power management for RTOS to the realm of communications and propose a **real-time dynamic power management scheme for the radio that is based on modulation scaling**.

II.   COMMUNICATION ENERGY

A.   Basics

In order to investigate modulation scaling, we need to know how the energy consumption depends on the modulation level. In this work, we focus on Quadrature Amplitude Modulation (QAM) due to its ease of implementation and analysis [4]. However, our notions of scaling and power management are extendable to other modulation schemes as well. The performance of QAM in terms of Symbol Error Rate (SER) and Bit Error Rate (BER) is given by (1)-(2) [7].

$$SER = b.BER = 4 \bullet \left(1 - \frac{1}{2^{b/2}}\right) \bullet Q\left(\sqrt{3 \bullet \frac{SNR}{2^b - 1}}\right) \qquad (1)$$

$$SNR = \frac{P_S}{P_n} \bullet A \qquad (2)$$

The constellation size in number of bits per symbol is represented by b. The received Signal to Noise Ratio is denoted by *SNR* and defined as (2), where P_S is the transmit power and P_n is the noise power. The factor A contains all transmission loss components. We can solve the above expressions for the transmit power, resulting in (3). The variable Γ is called the 'gap' and, for uncoded QAM, is given by (4), as can be derived from (1).

$$P_S = \frac{\Gamma \bullet P_n}{A} \bullet (2^b - 1) \qquad (3)$$

$$\Gamma = \left(\frac{1}{3}\right) \bullet \left[Q^{-1} \left(\left(\frac{1}{4}\right) \bullet \left(1 - \frac{1}{2^{b/2}}\right)^{-1} \bullet b \bullet BER \right) \right]^2 \qquad (4)$$

P_n is a function of the symbol rate R_s and the noise power spectral density N_0 as in (5) [7]. The factor $\mathcal{U}$ takes into account other effects, such as filter non-idealities.

$$P_n = N_0 \bullet \mathcal{U} \bullet R_S \qquad (5)$$

As a result, we can simplify the expression of P_S to (6)-(7). Γ_S in (7) is the gap for a fixed system operation point (b, *BER*). To get a fair comparison, the system performance is kept constant. In practice, it makes sense to operate at a target BER. The gap Γ in (7) is thus only a function of b. Due to the $Q^{-1}(.)$ function in (4), Γ and G_s only weakly depend on b.

$$P_S = C_S \bullet G_S \bullet R_S \bullet (2^b - 1) \qquad (6)$$

$$C_S = \frac{N_0 \bullet \mathcal{U}}{A} \bullet \Gamma_S \qquad G_S = \frac{\Gamma}{\Gamma_S} \qquad (7)$$

B. Energy and Delay

The goal now is to **adjust b and R_s** to reduce the overall energy. The transmit power P_S (delivered mainly by the power amplifier), however, is not the only source of power spending by the radio. Electronic circuitry for filtering, modulation, upconverting, etc. contributes as well. Equations (8)-(9) express this component P_E for a system that can dynamically change the symbol rate [8]. Parts of the circuitry operate at a frequency that follows the instantaneous symbol rate, while other parts have a fixed frequency proportional to the maximum symbol rate. The proportionality factors are incorporated in C_A and C_B.

$$P_E = \left[C_E + C_R \bullet \frac{R_{S_{max}}}{R_S} \right] \bullet R_S \qquad (8)$$

$$C_E = C_A \bullet V^2 \qquad C_R = C_B \bullet V^2 \qquad (9)$$

The total power consumption is the sum of both the transmit and electronics power. We can express the energy to transmit one bit, E_{bit}, as (10). In this equation, T_{bit} is the time it takes to transmit one bit. The term ε accounts for circuitry that is always on. Since this presents a fixed energy offset, we ignore this term in the remainder of this paper.

$$E_{bit} = (P_S + P_E) \bullet T_{bit} + \varepsilon \qquad (10)$$

The goal is to minimize the energy per bit by choosing the correct values of b and R_s. For typical applications, we need to constrain the total delay a packet may incur, translating to a bound on T_{bit}. We summarize the optimization problem as:

$$\min \quad E_{bit} = \left[C_S \bullet G_S \bullet (2^b - 1) + C_E + C_R \bullet \frac{R_{S_{max}}}{R_S} \right] \bullet \frac{1}{b} \qquad (11)$$

$$T_{bit} = \frac{1}{b \bullet R_S} = T_{max} \qquad (12)$$

III. PERFORMANCE TRADEOFFS

Our evaluation of the performance tradeoffs is based on the values given in Table 1. These are extracted from [8], which describes the implementation of an adaptive QAM system. Since the system in [8] is designed for high speed rather than low-power applications, it is likely that applying dedicated circuit techniques can reduce these numbers. We only use them here as proof of concept.

Figures 1 and 2 depict E_{bit} and T_{bit} as obtained from (11) and (12) respectively. After evaluating the performance for varying constraints on the delay (i.e. varying T_{max}), it turns out that operating at the maximum R_S is preferable for any b. This is logical as a higher R_S results in both a lower T_{bit} and a lower E_{bit}. For practical systems, R_S should be chosen as high as possible, considering implementation issues and their associated power penalties. Varying the constellation size b is therefore the only option to trade off energy versus delay. In real implementations, b does not have an infinitesimal granularity but typically only takes on even integers, indicated by the black arrows in figures 1 and 2.

TABLE 1
SIMULATION SETTINGS

R_{Smax}	1 MHz	C_S (b=4)	10^{-7} J
BER	10^{-5}	C_E	8. 10^{-8} J
		C_R	10^{-7} J
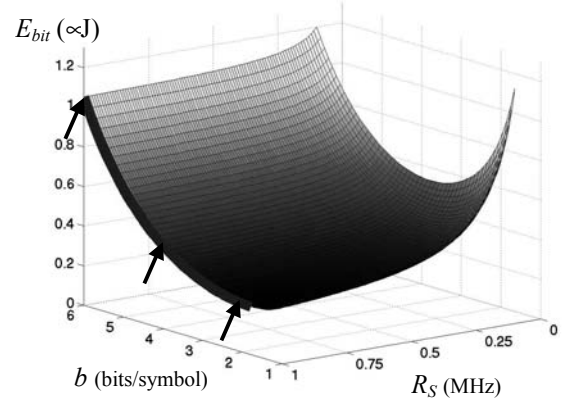
Fig. 1: Energy consumption for adaptive R_s system
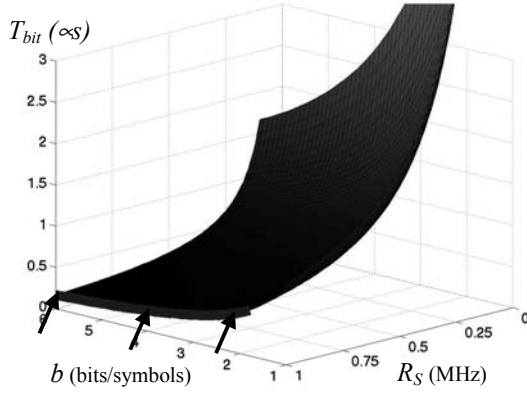
Fig. 2: Delay per bit

Since the optimal symbol rate is always the maximum one, the **variable symbol rate provision is not needed** for energy awareness. The circuitry that is described by the term with C_R is still present of course. We cannot simply remove this term, but modify (11) by setting R_{Smax} equal to R_S:

$$\min \quad E_{bit} = \left[C_S \bullet G_S \bullet \left(2^b - 1\right) + C_E + C_R \right] \bullet \frac{1}{b} \qquad (13)$$

In (13), E_{bit} is no longer a function of the symbol rate. Since a higher R_s results in a lower T_{bit}, it is again beneficial to operate at the highest symbol rate that can be implemented efficiently. Besides the advantage of lower delays, this also improves the capacity if the medium were shared. We can visualize the energy and delay curves by intersecting the surface in figures 1 and 2 with a plane at R_S = 1 MHz.

Note that there is a minimum in fig.1. In the region where the energy drops for a decreasing b, energy and delay can be traded off against each other. In analogy with voltage scaling techniques in digital circuits, we refer to this process as **modulation scaling**. Depending on the delay that is acceptable, the constellation size is lowered to meet that constraint with the minimum amount of energy. If this adaptation is performed on the fly, it results in **energy awareness**. It is useless to scale beyond the energy minimum point, as both delay and energy increase then. The position of this minimum depends on the relative values of C_S and (C_E $+C_R$). It turns out that, except for systems with a very short transmission range, operating at the smallest value of b is most energy efficient.

IV. REAL-TIME SCHEDULING

A. Real-Time Traffic Applications

In practical communication systems data is grouped into packets. We propose to add **a power management module to the radio**. Its task is to schedule the packet transmissions, while performing modulation scaling and at the same time maintaining the desired overall performance.

In this paper, we focus on **real-time traffic**, where each packet has a deadline by which it has to be sent. This model is valid for multimedia streams, such as audio or video. Another type of application that has real-time features is wireless sensor networks [3]. Envisioned uses for these networks of ultra-small autonomous devices are habitat monitoring, smart office and surveillance. A possible task of these networks is to periodically send updates on a condition (like the location of a target, the temperature of a room, etc.) to end users. These periodic updates can have different periods, depending on the application. The users tap into the network via gateway nodes, such as node G in fig. 3. These gateways receive multiple periodic traffic streams (e.g. one from A and one from B) that need to be scheduled on the (long-haul) link to the user in an energy efficient way.

Although we tackle the above scheduling problem in sensor networks, the power management scheme for real-time packet scheduling that we propose in this section is relevant for other classes of real-time traffic as well. In essence, we consider the general problem of **energy aware scheduling of real-time traffic streams via modulation scaling**.

B. Scheduling Basics

We characterize each real-time stream k by a **tuple (L_k,T_k)**, containing the maximum packet size and the time period. In each stream, the size of the individual packets might vary (e.g. in MPEG video or perceptual audio codecs). For sensor networks, this variation is due to changes in compression or the amount of sensor data. Each packet has to be sent before the next one arrives (so its deadline is its arrival time + T_k).

A similar problem of scheduling in the presence of deadlines has been studied extensively for RTOS. Energy awareness has been introduced there through voltage scaling. Techniques have been proposed that implement preemptive scheduling policies (i.e. tasks can be suspended and resumed later on) with the aim of minimizing the processor energy consumption [5][6]. However, in data transmissions, once a packet transmission has started, the scheduler should not interrupt the ongoing transmission. The condition of **non-preemptive scheduling** is the key distinction between packet scheduling on a link and task scheduling under an RTOS. Since all work on energy aware RTOS scheduling is preemptive [5][6], we cannot use these results. To the best of our knowledge, energy awareness has not been studied yet in the context of non-preemptive scheduling.
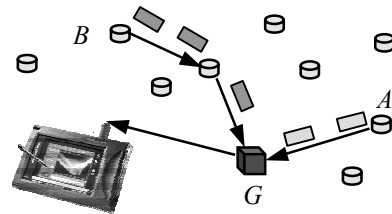
Fig. 3: Sensor network setup

Some useful results on the schedulability of task sets under non-preemptive scheduling (without scaling) are given in [9]. Finding the optimal schedule for a particular offset for each packet stream, where offset is defined as the first time a packet arrives, is an NP-complete problem [9]. Luckily, there exists a **schedulability condition (14)-(15)** (C_j is the transmit time for a packet of size L_j). When both (14) and (15) are satisfied, the set of N streams is schedulable for any offset. Otherwise, they might be schedulable for some, but there is at least one choice of offsets for which they are not. It can be shown that Earliest Deadline First scheduling (EDF) always results in a valid schedule when this condition is satisfied.

$$\begin{cases} \sum_{j=1}^{N} \dfrac{C_j}{T_j} = 1 & (14) \\[2em] .\ k = 1..N \quad .\ t_i = n \bullet T_i < T_k, \quad i = 1..(k-1) \\[1em] t_i = C_k + \sum_{j=1}^{k-1} \left\lfloor \dfrac{t_i}{T_j} \right\rfloor \bullet C_j - 1 \end{cases} \quad (15)$$

C. Energy-Aware Non-Preemptive Scheduling

Our goal is to make the non-preemptive real-time packet scheduling energy aware. An optimal scheduling routine would have to consider both the offsets and the variable packet sizes. This is too computationally intensive, since the problem was already NP-complete when only considering offsets. As a solution, we propose a practical algorithm, which consists of two steps.

Admission step: When a new stream is admitted to the system, we calculate a **static scaling factor α_{static}** for the system, assuming all packets are of maximum size. This factor is the minimum possible such that if the modulation setting for each packet would be scaled by it (which results in a uniform increase in C_j), the schedulability test (14)-(15) is still satisfied. In other words, it computes the slowest transmission speed for each packet at which all the packet streams are just schedulable. Fig. 4(a) depicts the original schedule with modulation b_{max} for three streams A, B and C. The deadline T of the previous packet is also the arrival time of the next packet of that stream. The slowdown in fig. 4(b) is indeed the maximum possible, since the first instance of packet B would otherwise miss its deadline T_B.

Adjustment step: During run-time, packets are scheduled using EDF. However, before the transmission starts the actual size of each packet is known, see fig. 4(c). We calculate an additional **scaling factor α_{dyn}** such that the transmission finishes when that of a maximum size packet would have, see fig. 4(d). Since step 1 assumed the maximum packet size, the schedulability is guaranteed. If the system would still be idle after the packet transmission, we stretch the transmission until the packet's deadline or the arrival time of a new packet (which is known due to the periodic nature of the traffic). We call this extra **scaling factor α_{strech}**. Fig. 4(e) illustrates that
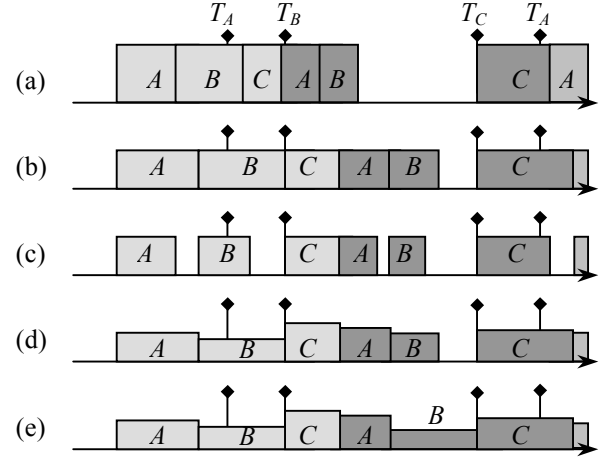
Fig. 4: Scheduling with modulation scaling

the second instance of packet B can indeed be stretched (while this factor is 1 for the other packets).

Finally, the scheduler combines all three scaling factors to get the overall modulation that is used for the current packet. For practical systems, the scaled constellation size b^* of (16) is rounded up to the closest allowable value.

$$b^* = b_{max} \bullet \alpha_{static} \bullet \alpha_{dyn} \bullet \alpha_{stretch} \quad (16)$$

D. Performance Evaluation

In order to evaluate the performance of our scheme, we have carried out a number of simulations. Unfortunately, for the sensor network application we focus on, no realistic numbers for the characteristics of data streams are available. We have based our simulations on general statistics, which we expect to be sufficiently relevant.

We select the maximum packet size L_k to be the same for each stream and equal to 400 bits. For each packet, its actual size is independently chosen from a uniform distribution between $\ell.L_k$ and L_k (where ℓ is a parameter we vary in our simulations). Furthermore, we choose R_s and b_{max} to be equal to 1 kHz and 6 bits/symbol respectively. All simulations are averaged over an adequate number of offsets.

Scenario 1: Five streams with different periods, listed in table 2, need to be scheduled. For these values, the total link utilization is rather high: 84% when all packets are of maximum size ($\ell = 1$). Fig. 5 plots the energy curves, normalized versus a scheme without scaling ($b = b_{max}$ at all times), of specific variants of our algorithm. When only using α_{static} in (16), there are no energy savings as the scaling factor is not enough to lower the modulation to next allowed level without compromising the schedulability.

TABLE 2
STREAM PERIODS

T_1	T_2	T_3	T_4	T_5
0.20 s	0.25 s	1.50 s	1.00 s	0.50 s

Relative energy
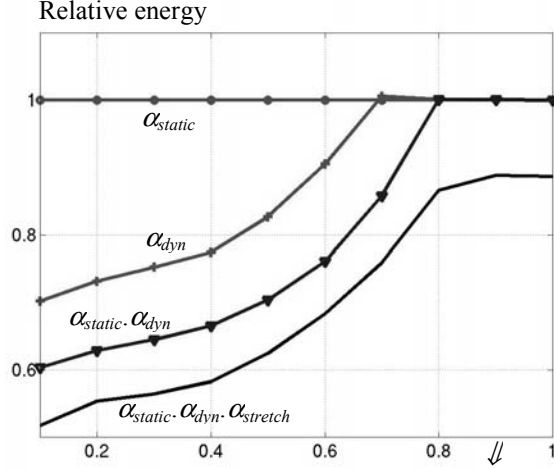
Fig. 5: Energy in scenario 1

However, adding α_{dyn} results in an energy reduction as the packet size variation increases ($\Downarrow$ decreases). We also note that the effect of α_{strech} is significant.

Scenario 2: Now, only the first three streams of table 2 are scheduled. In this case the maximum total link utilization is 64% and the normalized energy is shown in fig. 6. We note that in this case, α_{static} alone is large enough such that b can be uniformly set to 4 bits/symbol. Again α_{dyn} and $\alpha_{stretch}$ result in extra energy savings, although smaller.

These simulations illustrate that our dynamic power management scheme actually introduces **energy awareness with regard to two distinct phenomena**.

1. Variations in overall utilization are handled by the admission step in our algorithm. These are due to changes in number of streams, which are likely to occur over relatively large time scales. The transmission is uniformly slowed down via α_{static} to automatically adapt to these changes.

Relative energy
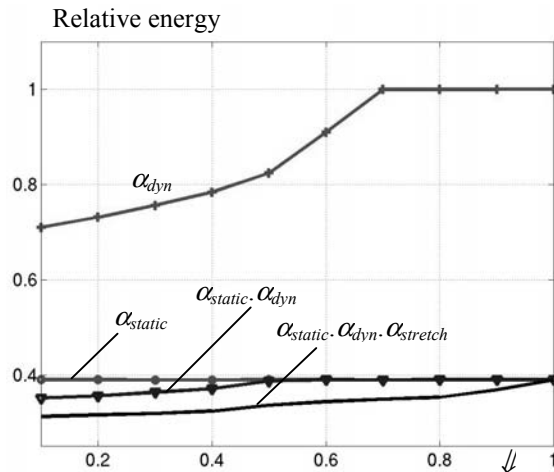
Fig. 6: Energy in scenario 2

2. Variations in individual packet sizes on the other hand occur at much smaller time scales. These cannot be handled during the admission of the streams, but are exploited in the adjustment step of our algorithm via α_{dyn} and $\alpha_{stretch}$.

V.   CONCLUSIONS

We have developed the concept of modulation scaling, which essentially allows us to trade off energy versus delay on the fly. As such, it enables the design of energy aware communication systems. We described how modulation scaling can be integrated into a dynamic power management scheme on the radio. As battery operated wireless communication devices become more proliferated, extending the principles of power management from the realm of embedded processors to embedded radios is indeed crucial.

To this end, we have presented energy aware EDF scheduling for real-time traffic streams. Our non-preemptive algorithm guarantees that no packet deadlines are missed, yet dynamically scales the transmission rate to ensure energy efficient operation. The adaptation leverages both long and short-term traffic dynamics, through a two-stage modulation scaling process.

REFERENCES

[1] Graybill, R., "DARPA Power Aware Computing/ Communication," *http://www.darpa.mil/ito/research/pacc /.*
[2] Pering, T., Bird, T., Brodersen, R., "The simulation and evaluation of dynamic voltage scaling algorithms," *ISLPED 1998,* pp. 76-81, August 1998.
[3] Sohrabi, K., Gao, J., Ailawadhi, V., Pottie, G., "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications Magazine*, Vol.7, No.5, pp. 16-27, Oct. 2000.
[4] Ue, T., Sampei, S., Morinaga, N., Hamaguchi, K., "Symbol Rate and Modulation Level-Controlled Adaptive Modulation/TDMA/TDD System for High-Bit Rate Wireless Data Transmission," *IEEE Trans. on Vehicular Technology*, Vol.47, No.4, pp. 1134-1147, Nov. 1998.
[5] Shin, Y., Choi, K., "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems," *DAC 1999*, New Orleans, LA, pp. 134-139, June 1999.
[6] Okuma, T., Ishihara, T., Yasuura, H., "Real-Time Task Scheduling for a Variable Voltage Processor," *ISSS 1999*, San Jose, CA, pp. 24-29, Nov. 1999.
[7] Proakis, J., "Digital Communications," *McGraw-Hill Series in Electrical and Computer Engineering*, 3rd Edition, 1995.
[8] Cho, K., Samueli, H., "A 8.75-MBaud Single-Chip Digital QAM Modulator with Frequency-Agility and Beamforming Diversity*," IEEE 2000 Custom Integrated Circuits Conference*, Orlando, FL, pp. 27-30, May 2000.
[9] Jeffay, K., Stanat, D., Martel, C., "On Non-Preemptive Scheduling of Periodic and Sporadic Tasks," *IEEE Real-Time Systems Symposium*, San Antonio, TX, pp. 129-139, Dec. 1991.

Adaptive Power-Fidelity in Energy-Aware Wireless Embedded Systems

Vijay Raghunathan, Paleologos Spanos, and Mani B. Srivastava
Networked and Embedded Systems Laboratory
Department of Electrical Engineering, UC Los Angeles
{vijay, paleolog, mbs}@ee.ucla.edu

Abstract

Energy aware system operation, and not just low power hardware, is an important requirement for wireless embedded systems. These systems, such as wireless multimedia terminals or wireless sensor nodes, combine (soft) real-time constraints on computation and communication with requirements of long battery lifetime. In this paper, we present an OS-directed dynamic power management technique for such systems that goes beyond conventional techniques to provide an adaptive power vs. fidelity trade-off.

The ability of wireless systems to adapt to changing fidelity in the form of data losses and errors is used to tradeoff against energy consumption. We also exploit system workload variation to proactively manage energy resources by predicting processing requirements. The supply voltage, and clock frequency are set according to predicted computation requirements of a specific task instance, and an adaptive feedback control machanism is used to keep system fidelity (deadline misses) within specifications.

We present the theoretical framework underlying our approach in the context of both a static priority-based preemptive task scheduler as well as a dynamic priority based one, and present simulation-based performance analysis that shows that our technique provides large energy savings (upto 76%) with little loss in fidelity (< 4%). Further, we describe the implementation of our technique in the eCos real-time operating system (RTOS) running on a StrongARM processor to illustrate the issues involved in enhancing RTOSs for energy awareness.

1 Introduction

Modern day wireless embedded systems, examples of which include mobile multimedia terminals with wireless LAN cards, 3G cellular phones, self-organizing ad-hoc network grids of wireless sensors, wireless toys and robots etc., involve an integration of high-performance computing and wireless communication capabilities, under real-time constraints. The battery operated nature of these systems requires them to be designed and operated in a highly energy efficient manner to maximize the battery lifetime.

In addition to using low-power hardware components [1, 2, 3], managing the various system resources in a power-aware manner i.e., Dynamic Power Management (DPM), can further reduce energy consumption considerably, thus increasing battery lifetime. This paper presents a system level pro-active DPM scheme that provides an adaptive trade-off between power consumption and operational fidelity by exploiting the key characteristics of wireless embedded systems listed below.

- **Most wireless systems are resilient to packet losses and errors.** The operating scenarios of these systems invariably have losses and errors associated with them (e.g., noisy wireless channel conditions lead to packet losses and errors). In these systems, the application layer is designed to be adaptive and tolerant to these impairments. Hence, these wireless systems fall under the category of "soft" real-time systems where a small number of task deadline misses are not catastrophic to system performance, but only lead to a small degradation in the user-perceived system fidelity. For example, a small percentage of packet loss is acceptable in mobile multimedia applications like audio or video decoding.

- **Wireless embedded systems offer a power-fidelity trade-off that can be exploited to suit application needs.** For example, the precision of the computation (e.g., quality of the audio or video decoding) can be traded off against the power consumed for the computation [21]. Also, the wireless channel induced error can be reduced by increasing the transmission power of the radio. Thus, these systems inherently have a power-fidelity control knob which can be fine tuned to suit application needs.

- **Wireless embedded systems have time varying computational loads.** Performance analysis studies have shown that for typical wireless applications (e.g., audio and video encoding/decoding, encryption/decryption etc.), the instance to instance task execution time varies significantly and is often far lower than the worst case execution time (WCET) [4]. Table 1 gives the WCET and best case execution time (BCET) for a few example benchmarks [4]. Note that, even though the execution times vary significantly, they depend on data values that are obtained from physical real-world signals (e.g., audio or video streams) and hence are likely to have some temporal correlation between them. This temporal correlation enables us to predict

Program	Description	BCET	WCET
DES	Data Encryption	73, 912	672, 298
DJPEG	JPEG decompression 128x96 color	12, 703, 432	122, 838, 368
FDCT	JPEG forward DCT	5, 587	16, 693
FFT	1024 point FFT	1, 589, 026	3, 974, 624

Table 1: Variation in execution times for a few tasks used in multimedia applications [4]

the runtime to a reasonable degree of accuracy. The variation in execution time offers additional potential for energy reduction which we exploit in the proposed work.

Due to the need for flexibility and adaptability to constantly evolving standards, a significant fraction of the functionality of wireless embedded systems is frequently implemented as software running under a real time operating system (RTOS) on a programmable processor platform. The RTOS is uniquely poised to efficiently implement system power management policies due to the following reasons. Since the RTOS co-ordinates the execution of the various application tasks, it has global information about the performance requirements and timing constraints of all the applications. Also, the RTOS can directly control the underlying hardware platform, tuning it to meet specific system requirements.

A commonly used DPM scheme is one in which unused system components are shutdown or sent into low-power states. Previous work on shutdown based power management has focussed on optimizing the state transition policy [5, 6, 7]. A detailed survey of system level shutdown based power management techniques is presented in [8]. An alternative DPM technique is Dynamic Voltage Scaling (DVS), where the voltage and operating frequency of the processor are changed dynamically during runtime in order to just meet the performance requirement. DVS based DPM, when applicable, has been shown to have significantly higher energy efficiency compared to shutdown based DPM due to the convex nature of the energy-speed curve [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21].

1.1 Paper overview and contributions

This paper presents a system level DVS based power management technique that exploits the previously described characteristics of wireless systems, namely error-tolerance, power-fidelity tradeoff, and time varying computational loads. The proposed technique consists of an **off-line** and an **online** component, that together exploit the task runtime variation by predicting task instance execution times and performing DVS accordingly. The few deadline misses that result from occasional erroneous prediction are not catastrophic to system performance due to the inherent error-tolerant nature of these wireless systems. Our algorithm uses a novel adaptive feedback mechanism to keep a tight check on the number of task deadline misses. This is done by

monitoring the deadline miss history, and accordingly adapting the prediction to be more aggressive or conservative.

We investigate a commonly used static priority assignment based scheduling scheme (Rate Monotonic priority assignment [22]) as well as a dynamic priority assignment based one (Earliest Deadline First priority assignment [22]) and show that our voltage scheduling strategy has a significantly higher energy efficiency compared to conventional WCET based schemes. Experiments indicate that the use of our technique results in **energy reductions of up to 76% (average of 54%) over a shutdown based technique, and up to 68% (average of 47%) over the WCET based technique of [15]**. We have implemented our DPM technique into the kernel of eCos [23], a commercial RTOS from Red Hat Inc., running on a StrongARM SA-1110 processor from Intel [24] that supports dynamic voltage and frequency scaling. We illustrate the various issues involved in enhancing an RTOS for energy awareness and describe how our implementation tackles these issues.

1.2 Related work

Initial work on DVS was in the context of a workstation-like environment [10, 11] where average throughput is the performance metric, and latency is not an issue since there are no real-time constraints. A technique to derive an off-line minimum-energy schedule for independent processes with deadlines was presented in [12]. Researchers have also proposed the concept of compiler-directed DVS [25] where the compiler sets the processor frequency and voltage with the aim of minimizing energy under real-time constraints. There have been several other techniques proposed for energy-efficient real-time task scheduling [13, 14, 15, 16, 17, 18, 19, 20]. However, all of them perform voltage scheduling assuming that the tasks in the system have hard deadline constraints. Further, most of them assume that every task instance executes for its WCET. Even the few techniques that take into account task runtime variation [15], leverage it only for processor shutdown and not during voltage scaling. Recently, a DVS technique that attempts to exploit task run-time variation was presented in [26]. However, this technique only targets hard real-time systems, and does not involve any pro-active DPM. Thus, it does not provide any tradeoff between power consumption and application fidelity.

The rest of this paper is organized as follows. Section 2 describes the background about real-time task scheduling and

presents examples to illustrate the power management opportunities that exist during task scheduling. Section 3 presents the CPU power model used. Section 4 discusses our adaptive power-fidelity tradeoff technique. Section 5 describes the simulation based performance analysis and presents simulation results. Section 6 describes our implementation setup (hardware and software), details the changes that have to be made to eCos to implement our power management scheme and presents the implementation results. Section 7 presents the conclusions.

2 Background and illustrative examples

We describe the basic scheduling approach adopted in our work, and present examples to illustrate the power management opportunities that arise due to the slack present in the system as well as dynamic variation of task execution times.

2.1 Task scheduling in RTOS

The task scheduler of an RTOS is responsible for scheduling a given set of tasks such that real-time constraints are satisfied. Schedulers differ in the type of scheduling policy they implement. A commonly used scheduling policy is Rate Monotonic (RM) scheduling [22]. This is a fixed-priority based preemptive scheduling scheme where tasks are assigned priorities in the inverse ratio of their time periods. Fixed priority based preemptive scheduling is commonly used in operating systems like eCos, WinCE, VxWorks, QNX, uC/OS etc., that run on wireless embedded devices such as handheld multimedia devices and wireless sensors. An alternative scheduling scheme is Earliest Deadline First (EDF) scheduling [22], where task priorities are assigned dynamically such that task instances with closer deadlines are given higher priorities. No matter which scheduling policy is used, the RTOS ensures that, at any point of time, the currently active task is the one with the highest priority among the ready to run tasks.

2.2 DPM opportunities during task scheduling

It has been observed in many systems that, during runtime, even if all task instances run for their WCET, the processor utilization factor is often far lower than 100%, resulting in idle intervals. This slack that inherently exists in the system due to low processor utilization is henceforth referred to as the **static slack**. It can be exploited for power management by statically slowing down the processor and operating at a lower voltage. While processor slowdown improves the utilization factor, excessive slowdown may lead to deadline violations. Hence, the extent of slowdown is limited by the schedulability of the task set under the Rate Monotonic scheduling scheme at the reduced speed. The following example illustrates the use of static slowdown and voltage scaling to reduce energy consumption by exploiting the static slack present in the system.

Example 1: Consider a simple mobile multimedia terminal shown in Figure 1. The system receives real-time audio and
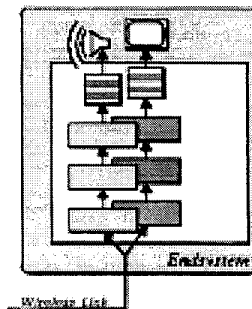
Figure 1: Mobile multimedia terminal used in Examples 1 and 2.

Task	Time Period	WCET	Deadline
Audio decoding	60	10	60
Protocol processing	70	15	70
Video decoding	120	40	120

Table 2: Task timing parameters for Examples 1 and 2

video streams over a wireless link, and plays them out. The three main tasks that run on a processor embedded in this system are protocol processing, audio decoding, and video decoding. The timing parameters for these tasks are given in Table 2. The resulting schedule for the time interval [0, 120] when this task set is scheduled on a single processor using the Rate Monotonic priority assignment scheme [22], is shown in Figure 2(a). It can be seen from the figure that the system is idle during time interval [90, 120]. This slack can be utilized to lower the operating frequency and supply voltage, thereby reducing the energy consumption. Figure 2(b) shows the schedule for the same task set with the processor slowed down by a factor of $\frac{4}{3}$. As seen from the figure, processor slowdown leads to elimination of slack. Note that any further reduction in processor speed will result in the video decoding task missing its deadline. So, for minimum energy operation while satisfying deadline constraints, the processor's operating frequency can be lowered by a factor of $\frac{4}{3}$ and the supply voltage can be lowered from 5V to 4.08V (using the equations presented in Section 3). The energy consumption which is quadratically dependent on the supply voltage is reduced by 33.42%. ■

Static slack is not the only kind of slack present in the system. During runtime, due to the variation in the task instance execution times, there is an additional slack created when a task instance finishes executing before its WCET. This slack that arises due to execution time variation is henceforth referred to as **dynamic slack**. Dynamic slack can be exploited for DPM by dynamically varying the operating frequency and supply voltage of the processor to extend the task execution time to its WCET. The faster a task instance executes, the more its power management potential. However, to realize this potential, we need to know the execution time of each task instance. As mentioned in Sec-

105
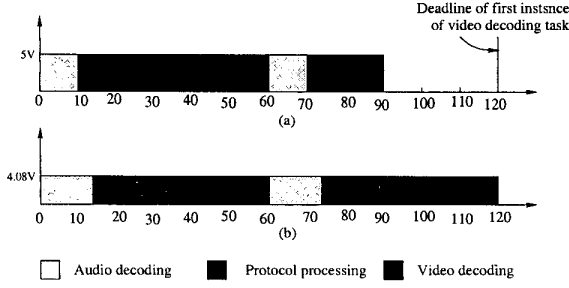
Figure 2: Task execution schedule for Example 1: (a) Original (b) Statically optimized

tion 1, the temporal correlation between the instance to instance execution times, enables us to predict the runtime of a task instance based on recent execution history. The following example illustrates how energy efficiency can be increased by utilizing the dynamic slack.

Example 2: Consider the optimized schedule for the task set of Example 1 shown in Figure 2(b). Figure 3(a) shows the resulting schedule when the first instance of video decoding requires only 20 time units to complete execution. As seen from the figure, there is a slack created due to the execution time variation of the video decoding. This slack can be utilized to reduce energy by dynamically reducing the processor frequency and supply voltage during the execution of the first instance of the video decoding task. If the frequency were to be dropped by a factor of 2 over the already statically optimized value, the slack gets filled up again as shown in Figure 3(b). The voltage can then be decreased to 2.69V (using the delay-voltage equation given in Example 1). This gives us an additional energy savings of 56% for that particular task instance, over the statically optimized value. ∎
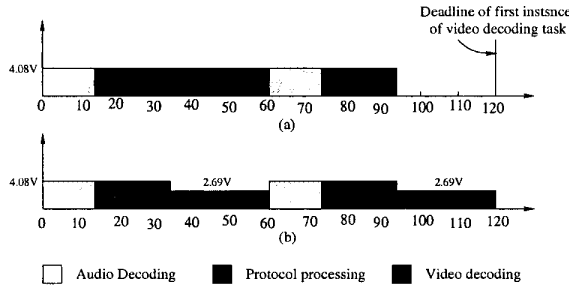
Figure 3: Task execution schedule for Example 2: (a) Statically optimized (b) After dynamic slowdown

The above examples illustrated the power management opportunities present during real-time task scheduling. We next describe our system model and then present our power-aware scheduling algorithm that exploits these opportunities to result in large energy savings.

3 CPU power model

The power consumed by the processor is dominated by the switching component which is given by $P_{switching} = C_{eff} \times V_{dd}^2 \times f$ where C_{eff} is the total switched capacitance, V_{dd} is the supply voltage, and f is the system clock frequency. The gate delay in the system is given by $Delay = \frac{k \times V_{dd}}{(V_{dd}-V_{th})^2} \propto \frac{1}{f}$ where V_{th} is the threshold voltage, and k is a technology dependent constant. The maximum voltage of the system was set to be $V_{ref} = 5$ volts for the simulation based analysis. The normalized speed of the processor at maximum frequency, and the normalized power consumption at maximum frequency and maximum supply voltage were both set to be 1. By combining the above equations, and eliminating V, the normalized power consumption (P) as a function of the normalized speed (S) is obtained as :

$$P(S) = 0.248 * S^3 + 0.225 * S^2 + 0.0256 * S + (0.014112 * S^2 + 0.0064 * S) * \sqrt{311.16 * S^2 + 282.24 * S}$$

for $V_{th} = 0.8V$. We use this equation to calculate the power consumption of the system for a given normalized operating speed. However, our implementation (described in Section 6) uses a power model derived from actual measurements [27].

3.1 Variable voltage supply

The variable supply voltage is generated using DC-DC switching regulators in the power supply. Efficient DC-DC regulators with fast transition times were reported in [28, 29]. Complimentary to the supply voltage variation is the accompanying variation of the system clock frequency. When the clock frequency is altered, the PLL present in the clock generator circuitry takes a finite amount of time to settle at its steady state value. In our simulation framework, we account for these transition overheads, by setting the transition time for the voltage change and associated frequency change to be 10 microseconds [17]. Further, the processor is stalled for the duration of the frequency change, resulting in some amount of wasted energy. We account for this overhead as well in our energy calculations.

4 Algorithm

At the core of our system model is the runtime scheduler that schedules applications as concurrent tasks to run on the processor. We consider a priority based scheduling model where each task is given a priority and higher priority tasks are always executed before lower priority ones. The priority assignment is either done statically, in the case of Rate Monotonic scheduling [22] or dynamically, in the case of the Earliest Deadline First scheduling [22]. Since other tasks may arrive when a particular task instance is being serviced, there is a queue to store the ready to run tasks. The preemptive nature of the system ensures that if at any instant of time a higher priority task than the active one arrives, the active task is preempted and sent to the ready to run queue and the higher priority task starts executing. Whenever

106

a task finishes executing, the highest priority task in the ready queue is selected to be executed next. Task instances that do not finish executing by their deadline are killed.

The pseudo-code of our algorithm is shown in Figures 4 and 5. The crucial steps of our algorithm, namely computation of the static and dynamic slowdown factors and the voltage variation policy, are described in the following subsections.

4.1 Static slowdown factor computation

This is the off-line component of our algorithm and involves a static analysis of the target task set. It is executed whenever a new task (*e.g.*, audio/video decoding) enters the system and registers itself with the scheduler. Given the task set to be scheduled, the procedure COMPUTE_STATIC_SLOWDOWN_FACTORS performs a schedulability analysis [30] and computes the minimum operating frequency for each task, at which the entire task set is still schedulable. Every task is slowed down uniformly till one or more tasks reach their critical point (*i.e.*, they are *just* schedulable). Further scaling of any task with higher priority than any of the critical tasks will render at least one critical task unschedulable. Therefore, only those tasks, if any, with lower priority than any of the critical tasks can be scaled further without affecting the schedulability of the task set. The procedure continues till there is no further scaling possible while satisfying all task deadlines. This procedure gives us a reduced clock frequency for each task such that the entire task set is just schedulable. The online component of our algorithm builds upon this static slowdown to perform dynamic frequency and voltage scaling to maximize the energy savings.

```
Procedure COMPUTE_STATIC_SLOWDOWN_FACTORS
Inputs: Time_Periods[ ], WCETs[ ], Deadlines[ ]
Outputs: Static_Slowdown_Factors[ ]
{
SET S //Tasks that can be further slowed down
SET S1
Current_Scaling_Factor = 1;
While (Scalable_tasks ! = φ) {
  f = Scale_WCETs(Periods[ ], WCETs[ ], Deadlines[ ]);
  S1 = Tasks that will miss deadlines
      with further scaling;
  Current_Scaling_Factor = Current_Scaling_Factor * f;
  For each task i in S
    Static_Slowdown_Factor[i] = Current_Scaling_Factor;
  S = All tasks with priority < Lowest priority
      of tasks in S1;
  }
}
```

Figure 4: Pseudo-code for the off-line component of the proposed algorithm

4.2 Dynamic voltage scaling

The online component of our algorithm invokes the COMPUTE_DYNAMIC_SLOWDOWN_FACTOR procedure to dynamically alter the supply voltage and operating frequency of the system during task scheduling in accordance with the recent task execution history. The procedure is called each time a new task instance starts executing, and based on an estimate of the task instance execution time, the processor speed and voltage are set accordingly.

```
Procedure COMPUTE_DYNAMIC_SLOWDOWN_FACTOR
Inputs: WCET, Deadline_Miss_History,
        Execution_History
Outputs: Dynamic_Slowdown_Factor
{
P = PREDICT_TIME(Execution_History);
α = ADAPTIVE_FACTOR(Deadline_Miss_History, α);
P = P * α;
Dynamic_Slow_Factor = WCET / P;
}


Procedure PREDICT_TIME
Input: Execution_History
Output: Predicted_time
{
If (Elapsed_time_for_task_instance = 0)
  Predicted_time = $\frac{\Sigma_1^N (Runtime\ of\ kth\ task\ instance)}{N}$;
  //N is the number of past instances being monitored
Else
  Predicted_time = Expected value of execution time
      distribution from Elapsed_Time to WCET;
}


Procedure ADAPTIVE_FACTOR
Input: Deadline_Miss_History, Adaptive_Factor
Output: Adaptive_Factor
{
x = Number of deadline misses in past M instances;
If (x ≥ T1) Adaptive_Factor + = I;
If (x ≤ T2) Adaptive_Factor - = D;
}
```

Figure 5: Pseudo-code for the online component of the proposed algorithm

4.2.1 Runtime prediction strategy

A novel feature of our algorithm is that it uses a predictive scheme that exploits the temporal correlation between the instance to instance task execution times to determine the run times of individual task instances. The runtime of a task instance is predicted as the mean of the execution times of a fixed number of previous task instances. An execution history database is maintained for each task, that is updated every time a task instance finishes executing or is killed owing to a deadline miss. This is a very simple predictor that places an extremely light computational load on the scheduler. The use of a more complex predic-

tor will certainly improve prediction accuracy, and therefore the performance of our algorithm. However, this will significantly increase the computational burden on the scheduler, leading to non-negligible scheduler overhead.

4.2.2 Improving prediction accuracy

In order to improve prediction accuracy while retaining simplicity, we extend the prediction strategy to use *conditional prediction* as described below. Every time a task instance is preempted, the predicted remaining time for that task instance is recalculated as the expected value of the past execution history distribution from the already elapsed time to the WCET of the task. This gives the predicted remaining time of the task, given that it has already run for the elapsed time. This improves the prediction accuracy and reduces the probability of under-prediction, in turn reducing the probability of missed deadlines.

4.2.3 Adaptive power-fidelity tradeoff

To further reduce the number of missed deadlines, we introduce a novel adaptive feedback mechanism into the prediction process. The recent deadline miss history is monitored, and if the number of deadline misses is found to be increasing, the prediction is made more conservative, reducing the probability of further deadline misses. Similarly, a low/decreasing deadline miss history results in more aggressive prediction in order to reduce energy consumption. The prediction scheme thus becomes adaptive to a recent history of missed deadlines, becoming more conservative in response to deadline misses and becoming more aggressive if no deadline misses occur over the past few instances. This adaptive control mechanism ensures that the number of deadline misses (which is representative of system fidelity) is kept under tight check at all times.

4.2.4 Voltage variation policy

The pre-computed static slowdown factor for each task is augmented with a dynamic slowdown factor for the specific task instance that is about to start executing. The dynamic slowdown factor is computed by stretching the task instance's predicted execution time to reach its WCET. The product of the static and dynamic factors is the final slowdown factor. The processor's operating frequency is then decreased by this factor, the corresponding supply voltage is set, and execution of the task instance begins. This dynamic slowdown spreads the execution to fill otherwise idle time intervals, enabling operation at a lower supply voltage and frequency, and resulting in reduced energy consumption.

5 Simulation based analysis

The proposed DVS based adaptive power management scheme was implemented and simulated using PARSEC [31], a C based discrete event simulator. The system model for the simulation is

shown in Figure 6. The implementation consisted of two parallel communicating entities representing the RTOS, and the task source. In addition to performing task scheduling, the RTOS entity also maintained and monitored the task instance execution history and the deadline miss history. The task generator, periodically generated task instances with run times according to a trace or a distribution, and passed them to the RTOS entity.
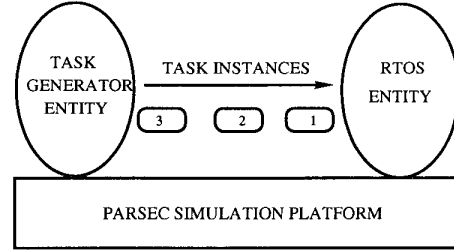
Figure 6: System simulation model in PARSEC

We performed simulations on three different task sets. The first two task sets (henceforth called *Task Set 1* and *Task Set 2*) are based on standard task sets used in embedded real-time applications [32, 33]. The third task set is a multimedia set that has two tasks - MPEG and audio decoding. For the first two task sets, task instance run times were generated from a random Gaussian distribution with Mean = (BCET + WCET) / 2 and Standard Deviation = (WCET - BCET) / 6. The actual run times for the multimedia tasks were derived from the discrete probability distribution for these given in [34]. We conducted three experiments on each task set. In the first experiment, only shutdown based DPM was employed and the operating voltage and frequency remain fixed at their maximum values. In the second experiment, we implemented the low-power scheduling technique of [15].

This scheme is a non-predictive one where the voltage scaling is based on the WCET. Finally, the proposed prediction based adaptive power management scheme was implemented and the power savings were recorded. For the first two task sets, each experiment was repeated while varying the BCET from 10% to 100% of the WCET in steps of 10% and the power consumption was calculated. Figures 7-9 present the results when RM scheduling is employed and show the power consumed in the various cases, for Task Set 1, Task Set 2 and the multimedia task set respectively. As can be seen from the figures, shutdown based
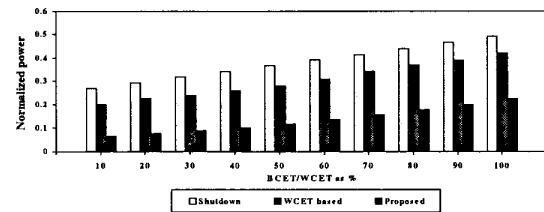
Figure 7: Normalized power dissipation of various DPM schemes for Task Set 1 under RM scheduling
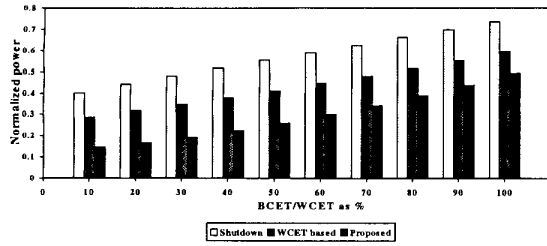
Figure 8: Normalized power dissipation of various DPM schemes for Task Set 2 under RM scheduling
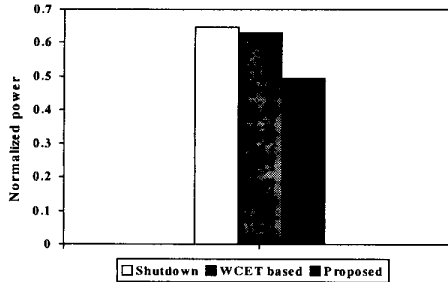
Figure 9: Normalized power dissipation of various DI schemes for the multimedia task set under RM scheduling

power management is the least energy efficient while the proposed DVS based adaptive power management technique results in a high energy savings. The proposed DPM technique results in energy savings of upto 76% (average of 54%) and upto 68% (average of 47%) over the shutdown based scheme and WCET based scheme respectively. The deadline misses that occur due to the predictive nature of the proposed scheme are plotted in Figure 10 for the first two task sets. In the case of the multimedia task set, for the particular runtime distribution used, the percentage of deadline misses was found to be 4%, which is acceptable in speech and video processing applications. Similar results were obtained when EDF scheduling was used. Figure 11 shows the power consumption for Task Set 1 when scheduled using the EDF policy and Figure 12 shows the corresponding deadline miss profile.

It is clear from the above results that the use of our adaptive prediction based DVS technique results in a large savings in the power consumed at the cost of an extremely small percentage of missed deadlines.

6 Implementation issues and experimental validation

We have implemented the proposed adaptive power-aware task scheduling scheme into the task-scheduler of the eCos operating system [23], which is a commercial RTOS from RedHat Inc. We used the Assabet development board [24] from Intel as the hardware platform for porting the eCos operating system. Figure 13 shows a picture of the Assabet development board. We next
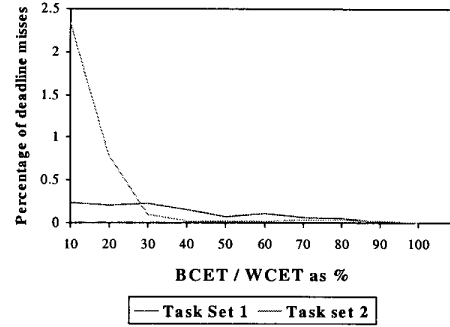
Figure 10: Percentage of deadline misses for the Task Set 1 and Task Set 2 under RM scheduling
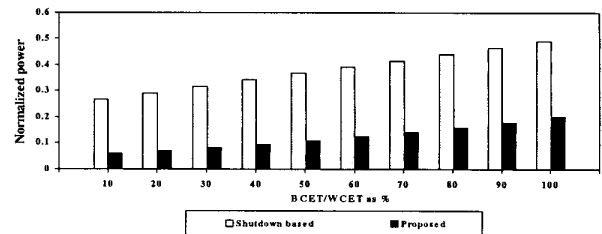
Figure 11: Normalized power dissipation of various DPM schemes for Task Set 1 under EDF scheduling

highlight some of the issues involved in implementing DVS on an RTOS, and describe our implementation in detail. We present experimental results obtained from our implementation that validate and prove the effectiveness of the proposed DPM technique.

6.1 The StrongARM SA-1110 processor

The Intel StrongARM SA-1110 processor is a high-performance, low-power processor for portable wireless multimedia devices. The StrongARM SA-1110 supports dynamic voltage and frequency scaling. The core clock frequency is configured by software through the core clock configuration field in the power manager phase-locked loop configuration register (PPCR). It can operate from 59MHz to 221MHz, with a corresponding core supply voltage from 0.8V to 1.5V. Voltage scaling has to be done externally using a DC-DC converter. The processor displays a dynamic range of 2X in energy per operation [35] which is indicative of the significant energy savings that can be obtained through the use of DVS. The following characteristics of the SA-1110 have a direct impact on the energy savings obtained.

- **Discrete permissible operating frequencies:** The SA-1110 can operate only at 12 discrete frequency values, between 59MHz and 221MHz. Thus, frequency hopping can only be done in discrete steps and not in a continuous manner. This leads to a small decrease in the extent of DVS, and hence the energy savings.

109
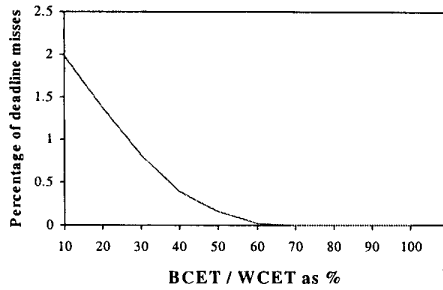
Figure 12: Percentage of deadline misses for the Task Set 1 under EDF scheduling
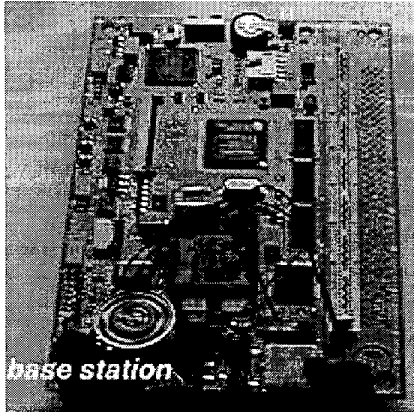
Figure 13: Photograph of the Assabet development board [25]

- **Time overhead associated with frequency changes:** Every time a frequency change is effected, the system clock is frozen, and the processor is stalled for 150 microseconds while the on-board PLL locks to the new frequency. During this frequency transition interval, the processor consumes some energy, but this is very small since the system clock is frozen, and so the switching power is eliminated. Newer processors such as Intel's XScale [36], and Transmeta's Crusoe [37] have a much smaller time overhead associated with frequency changes (around 30 microseconds).

- **Energy loss in the DC-DC converter:** During supply voltage conversion process, there is some energy loss in the DC-DC converter. In recent years, several high efficiency DC-DC converters have been proposed to power variable voltage systems [28, 29]. In our implementation, we do not actually power the SA-1110 with a variable voltage. This is because the SA-1110 development board is not equipped with a DC-DC converter. However, a board can easily be constructed to provide the required variable supply voltage. For our implementation, we scale the frequency and maintain a time stamped log of every frequency change, and the corresponding value of the supply voltage. We post-process the log to numerically calculate the power dissipation.

6.2 The eCos operating system

eCos is an open source real-time operating system for embedded applications. At the core of the eCos kernel is the runtime scheduler that defines the way the various threads run and provides the mechanisms by which they may synchronize. The scheduler which is responsible for the task execution schedule has to be enhanced so that it can alter the processor frequency and voltage while taking scheduling decisions, making the system power-aware. We next illustrate the enhancements that have to be made to eCos to incorporate our voltage scheduling scheme into it.
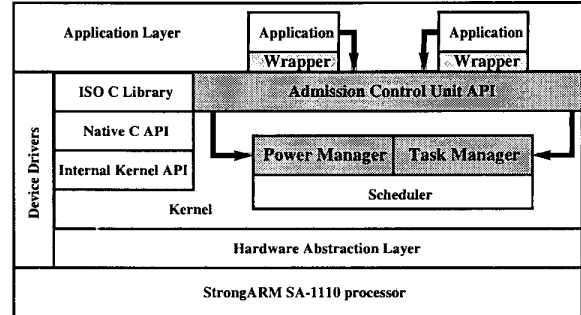
Figure 14: eCos Kernel block diagram showing the enhancements to be made

Figure 14 shows the changes that have to be made to eCOS in order to incorporate energy awareness into the task scheduling process. Our main objective while integrating power management into the RTOS kernel was to create a clean API module that acts as an interface between the user and the eCos kernel and enables the scheduler to make task scheduling decisions in an energy aware manner. Thus, any OS specific information (*e.g.* names of functions, sequence of function calls, object creation *etc.*) is hidden from the user. The API also acts as the Admission Control Unit and passes on user specified timing information to the kernel.

- **Application Wrapper:** Each application has to pass on its timing constraints to the runtime scheduler. For this purpose, a wrapper has to be added around every application in order to communicate with the Admission Control Unit.

- **Admission Control Unit:** Whenever a new task is introduced into the system, this unit performs a schedulability analysis, and decides whether to admit the new task or not. If the task is admitted, then this unit also implements the off-line component of our algorithm. The Admission Control Unit then passes on the static scaling factors, and the associated task timing information to the manager entities that are inbuilt into the kernel.

- **Task Manager:** Upon receiving task information from the admission control unit, the task manager generates the various threads and their associated runtime environment. The second important function of the task manager is deadline

110

monitoring. Under the current scheduling scheme of eCos, all task instances eventually run to completion. We introduce the notion of timing constraints through the task manager that monitors each task instance, and kills it if it violates its deadline. The task manager also performs a third function, namely that of dynamic priority-reassignment. When a new task enters the system after being approved by the admission control unit, the task manager locks the scheduler, creates the new task and its associated structures. It then reassigns priorities for all the tasks and re-registers them with the scheduler along with their new priorities. Control is then passed back to the scheduler to continue task execution.

- **Power Manager:** The power manager is responsible for all the DPM decisions made during scheduling. The power manager maintains all the timing information that is used to decide the operating voltage and frequency (history of task execution times, recent deadline miss information *etc.*). It also implements the runtime predictor based on the timing information. During a context switch, the power manager computes the task instance's scaling factor, and translates it into a valid frequency, which is then written into the PPCR register.
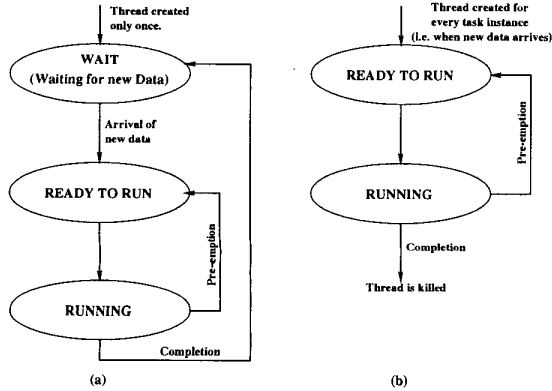
Figure 15: Thread implementation alternatives

6.2.1 Implementation architecture

Figure 15 shows two alternative thread implementation architectures. In the first architecture, shown in Figure 15(a), the thread corresponding to each application is created only once, when the task initially enters the system. The thread enters a WAIT state, where it waits for the arrival of new data (*i.e.* next task instance). The arrival of new data sends the thread into the READY_TO_RUN state. When picked by the scheduler for execution, the task then enters the RUNNING state. After completing one iteration of its computation (corresponding to one task instance), the thread goes back to the WAIT state and awaits new data. Thus, the same thread is re-initialized when new task instances arrive. In contrast, in the architecture shown in Figure

15(b), each task instance results in the creation of a new thread. The thread performs the required computation and is then killed along with its runtime environment. Upon the arrival of the next task instance, a new thread is spawned off and the process continues. For our implementation, we used the architecture shown in Figure 15(b), primarily due to ease of implementation.

6.3 eCos implementation results:
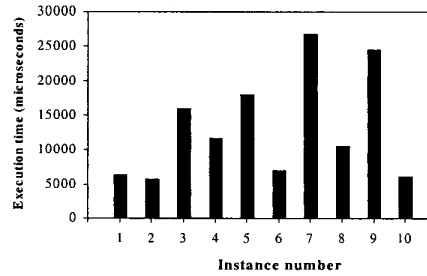
Figure 16: Execution time for various task instances of MARS encryption algorithm

We performed experiments on two application tasks running under eCos on the SA-1110, a speech decoder (ADPCM), and an encryption algorithm (MARS). These tasks represent typical applications that run on mobile multimedia embedded systems. The instance to instance execution time variation for the encryption task while running at the maximum operating frequency of 221Mhz is shown in Figure 16. We ran the two tasks concurrently on the SA-1110, and computed the power consumption. We used a power model based on actual measurements for computing the power consumption of the StrongARM processor. Figure 17 shows the power consumption of the StrongARM, plotted as a function of the operating frequency. This plot is obtained from actual current and voltage measurements reported in [27] for the StrongARM SA-1100 processor. Figure 18 shows the relative power consumption of the two schemes. As seen in the figure, our technique results in significantly higher energy savings than the shutdown based power management scheme.
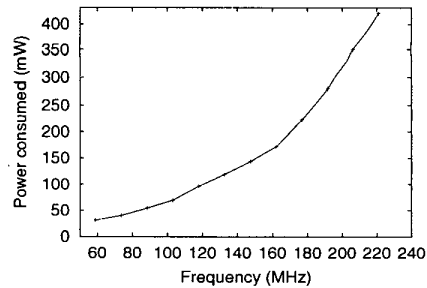
Figure 17: Power consumption as a function of operating frequency for the StrongARM processor [27]

111
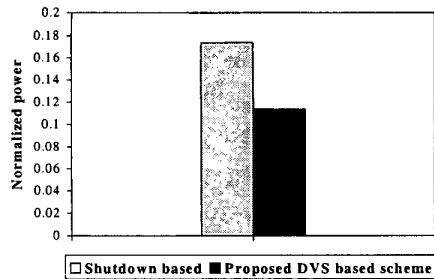
Figure 18: Comparison of normalized power consumption for the shutdown based scheme and proposed DVS based scheme running on the StrongARM SA-1110

7 Conclusions

A system level DVS based adaptive power management scheme was proposed for wireless embedded systems. The proposed technique exploits the low processor utilization factor, instance to instance variation in task execution times, and the tolerance to missed deadlines of wireless embedded systems to provide an adaptive power-fidelity tradeoff. The technique has been incorporated into the kernel of the eCos RTOS to result in a power-aware OS that reduces the energy consumption through aggressive DVS. Some ideas for future extensions include:

- **Analysis of adaptive scheme:** Our adaptive scheme has various parameters built into it whose optimal choice needs to be investigated.

- **Extending power management concepts to communication devices such as radios:** Radios work in a similar scenario where they are not needed to function all the time, and their workload varies with time too. So, the same kind of DPM techniques can be applied to radios as well, to minimize the energy consumption of wireless data transmission.

Acknowledgements

References

[1] A. P. Chandrakasan and R. W. Brodersen, *Low Power CMOS Digital Design*. Kluwer Academic Publishers, Norwell, MA, 1996.

[2] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, Norwell, MA, 1998.

[3] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, Norwell, MA, 1997.

[4] Y. -T. S. Li, and S. Malik "Performance analysis of embedded software using implicit path enumeration" in *IEEE Trans. on CAD*, vol. 16, iss. 12, pp. 1477-1487, Dec. 1997.

[5] M. B. Srivastava, A. P. Chandrakasan and R. W. Brodersen "Predictive system shutdown and other architectural techniques for energy efficient programmable computation" in *IEEE Trans. on VLSI Systems*, March 1996.

[6] T. Simunic, L. Benini, P. Glynn and G. De Micheli "Dynamic Power Management for Portable Systems" in *Proc. MOBICOM 2000*, Aug. 2000.

[7] G. A. Paleologo, L. Benini, A. Bogliolo, and G. De Micheli "Policy optimization for dynamic power management in *IEEE Trans. on CAD*, pp. 813-833, June 1999.

[8] L. Benini, A. Bogliolo, and G. De Micheli,"A survey of design techniques for system-level dynamic power management," in *IEEE Trans. on VLSI Systems*, vol. 8, iss. 3, pp. 299-316, June 2000.

[9] L. S. Nielsen and J. Sparso "Low-power operation using self-timed circuits and adaptive scaling of supply voltage" in *Proc. Int. Wkshp. Low Power Design*, pp. 99-104, April 1994.

[10] K. Govil, E. Chan, and H. Wasserman "Comparing algorithms for dynamic speed-setting of a low-power CPU" in *Proc. MOBICOM*, pp. 13-25, Nov. 1995.

[11] M. Weiser, B. Welch, A. Demers and S. Shenker, "Scheduling for reduced CPU energy" in *Proc. First Symp. on OSDI*, pp.13-23, Nov. 1994.

[12] F. Yao, A. Demers and S. Shenker, "A scheduling model for reduced CPU energy" in *Proc. Annual Symp. on Foundations of Computer Science*, pp.374-382, Oct. 1995.

[13] I. Hong, M. Potkonjak and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processors" in *Proc. ICCAD*, pp.653-656, Nov. 1998.

[14] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors, in *Proc. ISLPED*, pp.197-202, Aug. 1998.

[15] Y. Shin, and K. Choi,"Power conscious fixed priority scheduling for hard real-time systems," in *Proc. Design Automation Conf.*, pp. 134-139, June 1999.

[16] T. D. Burd, T. A. Pering, A. J. Stratakos and R. W. Brodersen "A Dynamic Voltage Scaled Microprocessor System" in *IEEE Journal of Solid-State Circuits*, vol. 35, iss. 11, pp. 1571-1580, November 2000.

[17] T. A. Pering, T. D. Burd and R. W. Brodersen "The simulation and Evaluation of Dynamic Voltage Scaling Algorithms" in *Proc. IEEE ISLPED*, pp. 76-81, August 1998.

[18] T. A. Pering, T. D. Burd and R. W. Brodersen "Voltage scheduling in the lpARM microprocessor system" in *Proc. IEEE ISLPED*, pp. 96-101, 2000.

[19] C. M. Krishna and Y. H. Lee, "Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems in *Proc. IEEE RTAS*, pp. 156-165, 2000.

[20] J. Luo and N. K. Jha, "Power-conscious Joint Scheduling of Periodic Task Graphs and Aperiodic Tasks in Distributed Real-time Embedded Systems in *Proc. IEEE ICCAD*, pp. 357-364, 2000.

[21] A. P. Chandrakasan, et. al. "An Architecture for a Power-Aware Distributed Microsensor Node" in *IEEE Wksp. on Signal Processing Systems (SiPS)*, October 2000.

[22] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment" in *Journal of ACM*, vol. 20, pp. 46-61, Jan. 1973.

[23] www.redhat.com/embedded/technologies/ecos

[24] http://developer.intel.com/design/strong/

[25] D. Mosse, H. Aydin, B. Childers, and R. Melhem "Compiler-Assisted Dynamic Power-Aware Scheduling for Real-Time Applications" in *Wkshp. on Compilers and Operating Systems for Low-Power*, 2000.

[26] F. Gruian,"Hard Real-Time Scheduling for Low Energy using Stochastic Data and DVS Processor," in *Proc. IEEE ISLPED*, August 2001.

[27] A. Sinha and A. P. Chandrakasan "Jouletrack: A web based tool for software energy profiling" to be presented at *IEEE/ACM Design Automation Conf.*, June 2001.

[28] W. Namgoong and T. H. Meng, "A high-efficiency variable-voltage CMOS dynamic dc-dc switching regulator" in *Proc. ISSCC*, pp.380-381, Feb. 1997.

[29] V. Gutnik and A. .P. Chandrakasan, "Embedded power supply for low-power DSP" in *IEEE Trans. on VLSI Systems*, Vol.5, No.4, pp.425-435, Dec. 1997.

[30] A. Burns, and A. Wellings, "Scheduling", Chapter 13 in *Real-time Systems and Programming Languages (2nd edition)*. Addison-Wesley, 1996.

[31] http://pcl.cs.ucla.edu/projects/parsec/

[32] N. Kim, M. Ryu, S. Hong, M. Saksena, C. Choi, and H. Shin, "Visual Assessment of a real time system design: a case study on a CNC controller" in *Proc. IEEE RTSS*, Dec 1996.

[33] A. Burns, K. Tindell, and A. Wellings, "Effective analysis for engineering real-time fixed priority schedulers" in *IEEE Trans. on Software Engineering*, Vol.21, pp.475-480, May 1995.

[34] A. Kalavade and P. Moghe, "ASAP - A framework for evaluating run-time schedulers in embedded multimedia end-systems" in *Proc. ACM Multimedia Conf.*, Sept. 1998.

[35] A. P. Chandrakasan, et. al. "Low power wireless sensor networks" in *IEEE Intl. Conf. VLSI Design*, January 2001.

[36] http://developer.intel.com/design/intelxscale/

[37] http://www.transmeta.com

SIMULATING NETWORKS OF WIRELESS SENSORS

Sung Park
Andreas Savvides
Mani B. Srivastava

Networked Emebedded Systems Laboratory
Electrical Engineering Departments
University of California, Los Angeles
Los Angeles, CA 90095, U.S.A.

ABSTRACT

Recent advances in low-power embedded processors, radios, and micro-mechanical systems (MEMs) have made possible the development of networks of wirelessly interconnected sensors. With their focus on applications requiring tight coupling with the physical world, as opposed to the personal communication focus of conventional wireless networks, these wireless sensor networks pose significantly different design, implementation, and deployment challenges. In this paper, we present a set of models and techniques that are embodied in a simulation tool for modeling wireless sensor networks. Our work builds up on the infrastructure provided by the widely used ns-2 simulator, and adds a suite of new features and techniques that are specific to wireless sensor networks. These features introduce the notion of a sensing channel through which sensors detect targets, and provide detailed models for evaluating energy consumption and battery lifetime.

1   INTRODUCTION

The marriage of ever tinier and cheaper embedded processors and wireless interfaces with micro-sensors based on micro-mechanical systems (MEMS) technology has led to the emergence of *wireless sensor networks* as a novel class of networked embedded systems. Many interesting and diverse applications for these systems are currently being explored. In indoor settings, sensor networks are already being used for condition-based maintenance of complex equipment in factories. In outdoor environments, these networks can monitor natural habitats, remote ecosystems, endangered species, forest fires, and disaster sites.

The primary interest in wireless sensor networks is due to their ability to monitor the physical environment through ad-hoc deployment of numerous tiny, intelligent, wirelessly networked sensor nodes. Because of the large numbers of sensor nodes required, and the type of applications sensor networks are expected to support, sensor nodes should be small, tetherless, and low cost. Due to these requirements, networked sensors are very constrained in terms of energy, computation and communication. The small form factor requirement prohibits the use of large long lasting batteries. Low production costs and low energy requirements suggest the use of small, low power processors, and small radios with limited bandwidth and transmission ranges. The ad-hoc deployment of sensor nodes implies that the nodes are expected to perform sensing and communication with no continual maintenance and battery replenishment. The energy constraints call for power awareness, which in turn leads to additional tradeoffs. The high-energy costs associated with wireless transmission, made particularly severe for sensor networks because nodes with small antenna heights placed on the ground see $1/r^4$ wireless link path loss coupled with the ever reducing cost of processing has led to a the adoption of a distributed computing viewpoint for wireless sensor networks. Instead of simply sending the raw data (perhaps compressed) to a gateway node, in typical applications the nodes in wireless sensor networks perform computation for decision making within the network, either individually via techniques such as signature analysis or in local clusters using coherent combining of raw sensor signals (i.e. beam forming) or non-coherent combining of decisions (i.e. Bayesian data fusion). By performing the computation inside the network, communication may be reduced thus prolonging the network lifetime

We construct a versatile environment in which sensor networks can be studied. This environment employs a wide range of models to orchestrate and simulate realistic scenarios. Furthermore, since power consumption is also a key design factor, we emphasize power consumption and battery behavior models. First we create a set of sensor node models that are derived from the empirical power characterization of two different nodes representing two extremes; the WINS node (Rockwell Scientific Company LLC. 2001) from Rockwell Science Center and the Medusa node, an experi-

113
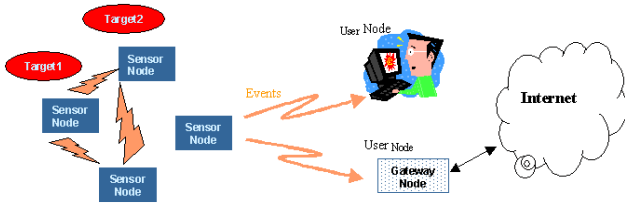
Figure 1: Sensor Network Scenario

mental prototype that we have constructed. These sensor node models are combined into the widely used event queue based network simulator, ns-2 (ns-2 Simulator 2001). By introducing the notion of *sensing channels* in our simulation environment and a flexible and highly parametrizable scenario generation tool, we can study the power consumption of sensor nodes by instrumenting complex sensor network scenarios in a detailed graphical environment.

2 RELATED WORK

Although sensor networks have recently received a lot of attention, there are still not many formal tools available for the systematic study of sensor networks. The work in (Ulmer 2001) presents a Java based simulator for sensor networks. This is an online simulator that can create and simulate simple topologies but does not have any explicit models for sensors or power management. Up to this point there is no publication on this work. On the network simulation, numerous simulators are currently available such as GloMoSim, OPNET and ns-2. These simulators provide great flexibility in the simulation of wireless ad-hoc networks at all layers. Despite their effectiveness, these tools are currently not equipped for capturing all the aspects of interest in sensor networks.

3 SIMULATION ARCHITECTURE OVERVIEW

We motivate our discussion with an example of a sensor network illustrated in figure 1. In this example, a set of wireless nodes equipped different sensor for monitoring natural habitat. The results of these sensors are processed within the network and the final sensing report is forwarded via wireless links to the gateway nodes that makes the results available on the internet. The main goal of our work is to recreate such scenarios in a versatile simulation environment where the behavior of the sensor network can be analyzed.

In our simulation environment, a typical sensor network scenario will consist of three types of nodes: 1) *sensor nodes* that monitor their immediate environment, 2) *target nodes* that generate the various sensor stimuli that are received by multiple sensor nodes via potentially many different transducers (e.g. seismic, acoustic, infrared) over different sensor channels; e.g. a moving vehicle generates

ground vibrations that trigger seismic sensors and sound that triggers acoustic sensors, and 3*) user nodes* that represent clients and administrators of the sensor network. Shown in figure 2, three type of node models make up the key building blocks of our simulation environment. The sensor nodes are the key active elements, and form our focus in this section. In our model, each sensor node is equipped with one wireless network protocol stack and one or more sensor stacks corresponding to different types of transducers that a single sensor node may possess. The role of the sensor protocol stacks is to detect and process sensor stimuli on the sensing channel and forward them to the application layer which will process them and eventually transmit them to a user node in the form of sensor reports. In addition to the protocol and sensor stacks that constitute the algorithmic components, each node is also equipped with a power model corresponding to the underlying energy-producing and energy-consuming hardware components. This model is composed of an energy provider (the battery) and a set of energy consumers (CPU, Radio, Sensors). The energy consuming hardware components can each be in one of several different states or modes, with each mode corresponding to a different point in performance and power space. For example, the radio may be in sleep mode, receive mode, or one of several different
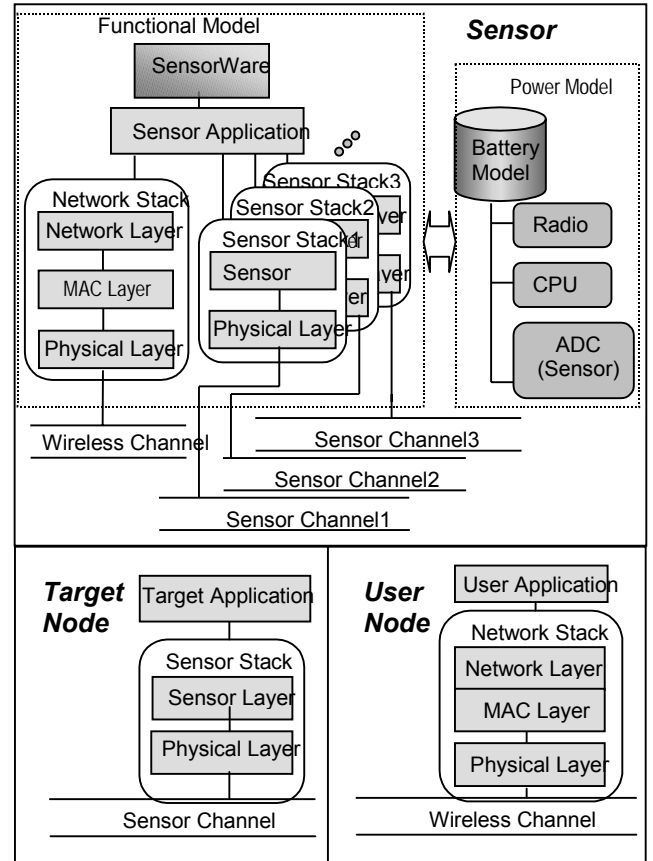
Figure 2: Sensor Node Model Architecture
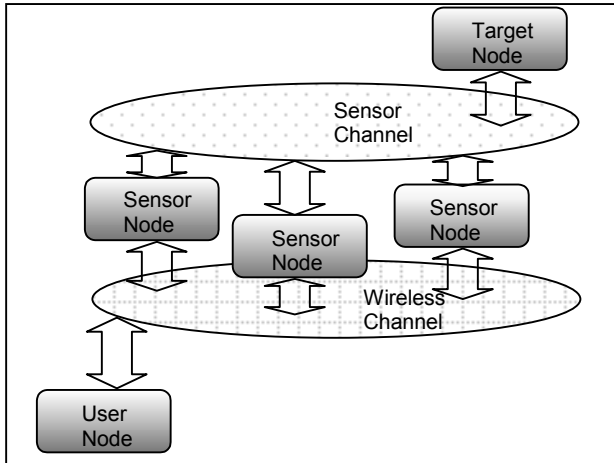
Figure 3: Sensor Network Model Architecture

transmit modes corresponding to different symbol rates, modulation schemes, and transmit power. Similarly, the CPU may be in sleep mode, or one of several different active modes corresponding to different frequency and voltage. The algorithms in the network and sensor stack control the change in mode of the power consumers. For example, the MAC protocol may change the radio mode from sleep to receive. In return, the performance of the algorithms may depend on the mode. For example, the time taken by the physical layer in the network protocol stack would depend on the data rate of the mode the radio is currently in. All of this is accomplished by having the algorithms in the network and sensor stacks issue mode change events to the power consumer entities, and having the algorithms read relevant parameter values from those entities. Algorithm-induced changes in the operating modes of power consuming hardware entities in turn affect the current drawn by them from the battery which delivers the power corresponding to the sum of current (or power) drawn by each power consumer. Internally, the battery entity depletes its stored chemical energy according to the efficiency dictated by the battery model.

Figure 3 illustrates how a typical sensor network will be constructed and simulated using our simulation environment. In figure 3, the wireless channel and sensor channel form separate communication mechanisms where events from different nodes are passed through. A typical scenario will involve a target node passing through a group of sensor nodes deployed in the field. As the target node moves around, it gives out sensor signal in the form of events through the sensor channel and each sensor node detects the events based on propagation model implemented in each node's sensor stack. When sensor nodes determine the sensor signals (events) are noteworthy, they transmit packets (also in the form of events) through the wireless channel destined to the user node.

By separating the sensor channel and the wireless channel, our sensor network model makes easier to simu-

late and analyze the operation of sensor network where the sensor signal detection events and wireless communication events can be received or transmitted concurrently. Moreover, by allowing sensor node to connect to multiple sensor channels, our simulation environment provides ability to analyze complex behaviors of sensor nodes' reaction to multiple sensor signals (i.e. seismic vibration, sounds, temperature, etc..) that can be detected all at the same time. In the following section, we discuss each components of the sensor node model shown in figure 2, and explain how we construct the model of different sensor nodes' components.

4 FRAMEWORK OF SENSOR NETWORK SIMULATION

4.1 Node Placement and Traffic Generation

In studying the performance of a wireless sensor network for a given application, a crucial element is the overall deployment scenario which includes the node placement topology, the radio ranges, the sensing ranges, the trajectories of the targets and resultant event traffics, and the trajectories of the user nodes and their query traffics. All these elements contribute to the different design trade-offs that can be made, and it is crucial to evaluate the effects of a new algorithm or protocol under diverse deployment scenarios.

To study such effects, we have developed a detailed scenario generation and visualization tool that enables us to construct detailed topologies and sensor network traffic. Our simulation environment enables us to assess the requirements of a sensor network under different circumstances by generating detailed scenario input to our simulations. This complements the scenario generation techniques provided in (ns-2 Simulator 2001) which are mainly targeted to ad-hoc wireless communication networks. Sensor node placement can vary depending on intended the task on the network. For example, to monitor wildlife in a forest, sensors may be uniformly distributed in the forest. If however, the sensor network is deployed for perimeter defense, then the sensors will most likely be distributed around a specified perimeter in a two dimensional gaussian distribution. In some other cases, the sensors may be manually placed according to the requirements of the user.

Besides placement, the traffic requirements may be even more diverse. Sensor network traffic can be classified into 3 main types: 1) *user-to-sensor* traffic, which is a result of user commands and queries to the network, 2) *sensor-to-user* traffic, which consists of the sensor reports to the user and 3) *sensor-to-sensor* traffic, which includes collaborative signal-processing of sensor events in the network before they are reported to the user. The last type of traffic is the most complex, and it depends on the sensing method.
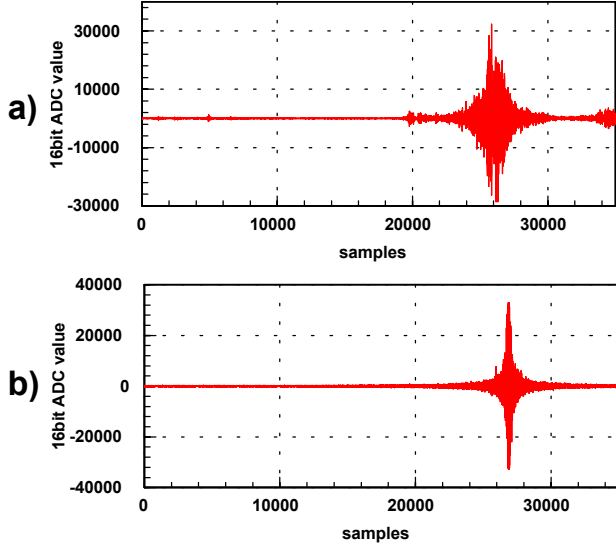
Figure 4: a) Real Target Seismic Signature
b) Simulated Target Seismic Signature

4.2 Sensor Stack and Sensor Channel

The *sensor stack* simulates how a sensor node generates, detects and processes sensor signals. In sensor node model (figure 2), the *sensor stack* is a signal sink that is responsible for triggering the application layer every time a sensing event occurs. Various trigger functions ranging from simple sensing schemes to elaborate signal processing functions can be implemented in the sensor stack. In target node model, the *sensor stack* acts as a signal source. The sensor stack of a target node will contain a signature that is unique to the type of target the target node is modeling. The signature is then transmitted through various mediums (ground, air, free space, water, etc..) as the target node moves around. figure 4a and 4b show a real and a simulated signature obtained from a seismic sensor triggered by ground vibration from a traveling vehicle.

In figure 4, the ground is the medium that transmits the vibrations to the seismic sensor. We refer to this medium as the sensor channel, a model of a medium which sensor events such as seismic vibration, sounds, or infrared signals are traveled through. The type of medium can differ based on the type of sensor being modeled (seismic, acoustic, infra red, ultrasonic). Moreover, depending on the medium being modeled, the propagation of signal can differ. For instance, a sound moving through the air will have different propagation as the same sound moving through the water. In order to incorporate all these different aspects of the sensor network in to our simulation, we implement a simple sensor stack and sensor channel model by modeling the target node as a gaussian source whose signal amplitude is modeled as a gaussian random variable with the mean equal to zero and the variance σ^2. As the

target travels through the sensor network, the target exerts the vibration signals (signal events) into the sensor channel periodically. The sensor channel then delivers this events to each sensor node's sensor stack and each sensor node adjusts the signal strength of the target based on sensor channels propagation model. The figure 4b demonstrates the signal strength variation as a target passes by a sensor node on a straight line. As the target approaches the sensor node, the signal strength increases, and as the target moves away, the signal attenuates rapidly. In this simulation the sensor signal was attenuated at a rate of 1/r where r is the distance between the target and the sensor.

4.3 Hardware Components Characterization

$$I_{sensor} = \frac{V_{test}}{R_{test}}$$

$$V_{Sensor} = V_{supply} - V_{test}$$

$$Power = I_{sensor} \times V_{sensor}$$

$$Energy = Power \times time$$
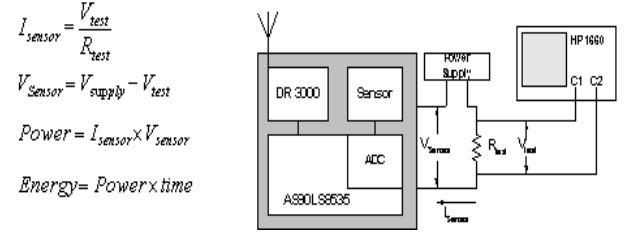
Figure 5: Power Measurement Configuration

Table 1: Experimental Node Current Consumption

Mode ID	CPU	Radio(OOK Modulation)	ADC	Total Current
1	Active 2.9mA	Tx-19.2kbps 5.2mA	On	8.1mA
2	Active 2.9mA	Tx-2.4kpbs 3.1mA	On	6.0mA
3	Active 2.9mA	Rx:4.1mA	On	7.0mA
4	Sleep 1.9mA	Sleep:5μA	On	1.9mA
5	Off 1μA	Sleep:5μA	Off	6μA

We construct our power models by performing measurements of the hardware power consumption using an HP 1660 oscilloscope, a bench power supply, and a high precision resistor. The measurement setup and power relationships are shown in figure 5. By characterizing each component of the sensor nodes we enable the simulated nodes to operate at different modes in which the power management schemes can switch different components on and off. Using the configuration in figure 5, the total current consumption of our experimental sensor node is obtained in Table 1. The measurements listed in Table 1 provide a better insight into the power consumption of the sensor nodes since the actual power consumption is oftentimes different from the typical values provided in the manufacturer data sheets depending on the mode of operation.

4.4 Battery Models

The Battery Model simulates the capacity and the lifetime of the sole energy source of the sensor node, the battery. In reality, battery behavior highly depends on the constituent materials and modeling this behavior is a difficult task. Although the battery can be viewed as a energy storage, the main goal of the sensor network is to increase the lifetime of the battery. Thus, in this section, we focus on how battery's capacity can be modeled based on the energy consumers' behavior. We propose 3 different types of battery models to study how different aspects of real battery behavior can affect the energy efficiency of different applications. The metrics that are used to indicate the maximum capacity of the battery is in the unit of Ah (Ampere*Hour). The metric is a common method used by the battery manufacturers to specify the theoretic total capacity of the battery. Knowing the current discharge of the battery and the total capacity in Ah, one can compute the theoretical lifetime of the battery using the equation ,

$T = \dfrac{C}{I}$, where *T*=battery lifetime, *C*=rated maximum battery capacity in Ah, and *I*=discharge current.

4.4.1 Linear Model

In Linear Model, the battery is treated as linear storage of current. The maximum capacity of the battery is achieved regardless of what the discharge rate is. The simple battery model allows user to see the efficiency of the user's application by providing how much capacity is consumed by the user. The remaining capacity after operation duration of time t_d can be expressed by the following equation.

$$\text{Remaining capacity (in Ah)} = C = C' - \int_{t=t_0}^{t_0+t_d} I(t)dt \qquad (1)$$

where *C'* is the previous capacity and *I(t)* is the instantaneous current consumed by the circuit at time *t*. Linear Model assumes that *I(t)* will stay the same for the duration t_d, if the operation mode of the circuit does not change (i.e. radio switching from receiving to transmit, CPU switching from active to idle, etc..) for the duration t_d. With these assumptions equation 1 simply becomes as the following.

$$C = C' - \int_{t=t_0}^{t_0+t_d} I(t)dt = C' - I \cdot t \Big|_{t_0}^{t_0+t_d} = C' - I \cdot t_d \quad (2)$$

The total remaining capacity is computed whenever the discharge rate of the circuit changes.
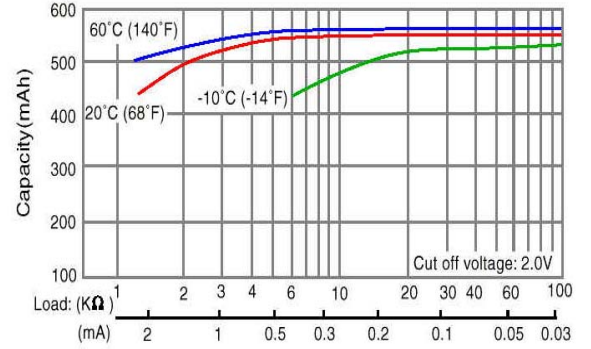
Figure 6: Capacity vs. Discharge Rate Curve for CR2354 (Matsushita Electric Corp. of America 2001)

4.4.2 Discharge Rate Dependent Model

While Linear Model assumes that the maximum capacity of the battery is unaffected by the discharge rate, Discharge Rate Dependent Model considers the effect of battery discharge rate on the maximum battery capacity. In [15] [16], it is shown that battery's capacity is reduced as the discharge rate increases. In order to consider the effect of discharge rate dependency, we introduce factor *k* which is the battery capacity efficiency factor that is determined by

the discharge rate. The definition of *k* is, $k = \dfrac{C_{eff}}{C_{max}}$,

where C_{eff} is the effective battery capacity and C_{max} is the maximum capacity of the battery with both terms expressed in unit of Ah. In Discharge Rate Dependent Model, the equation 1 is then transformed to the following.

$$C = k \cdot C' - I \cdot t_d \qquad (3)$$

The efficiency factor *k* varies with the current *I* and is close to one when discharge rate is low, but approaches 0 when the discharge rate becomes high. One way to find out corresponding *k* value is for different current value of *I* is to use the table driven method introduced in (Simunic 1999).

4.4.3 Relaxation Model

Real-life batteries exhibit a general phenomenon called "relaxation" explained in (Fuller 1994, Linden 1995, Chiasserini 1999). When the battery is discharged at high rate, the diffusion rate of the active ingredients through the electrolyte and electrode falls behind. If the high discharge rate is sustained, the battery reaches its end of life even though there are active materials still available. However, if the discharge current from the battery is cutoff or reduced during the discharge, the diffusion and transport rate

of active materials catches up with the depletion of the materials. This phenomenon is called relaxation effect, and it gives the battery chance to recover the capacity lost at high discharge rate. For a realistic battery simulation, it's important to look at the effects of relaxation as it has effect of lengthening the lifetime of the battery. For our simulation, we adapt the analytical model introduced in (Fuller 1994) which takes discharge rate as input and computes the battery voltage over the simulation duration.

5 EXAMPLE STUDY CASE

In this section we demonstrate some of the main capabilities of our tool by studying the performance of different battery models with various sensor node operation profile.

5.1 Low Rate/Low Power vs. High Rate/High Power

In this case study, we evaluate the battery consumption of our experimental sensor node by considering different operation profiles. In section 4.3, we have discussed how each component of our sensor node has different power consumption depending on its operation mode. In this section, we examine how the combination of the operation modes of different components affects the aggregate power consumption of the sensor node. The scenario involves two sensor nodes (a transmitter and a receiver) that are within the transmission range of each other (approximately 15 meters apart) where the transmitter needs to transmit a 2MB file to the receiver. For the purposes of our discussion we define 5 different operation modes for our experimental node shown in table 1. To examine the energy consumption and communication tradeoffs we evaluate 3 different data transmission policies.

1)19.2 kbps continuous transmission: The transmitter sends data at the highest data rate without any break. The transmitter will be operating in mode 1 and the receiver

will be operating in mode 3; 2) 2.4 kbps continuous transmission: With lower data rate the sender can transmit at a lower power level to reach the receiver. The transmitter will be operating in mode 2 and the receiver will be operating in mode 3; 3) 19.2 kbps pulse transmission: The transmitter sends data intermittently at the highest power level. While the transmitter is not transmitting, the transmitter puts the CPU and Radio to sleep. The transmitter power cycle its component by transmitting one 60 byte packet at 19.2 kbps for .025 sec and sleeps for .125 sec until all the data is received by the receiver. The transmitter will be switching between modes 1 and 4, and the receiver will be switching between mode 3 and 4.

Figure 7a shows the effect on each battery model capacity after the 2 MB data transfer for the three transmission methods described above. This experiment was performed for all three battery models described in section 4.4. Initially, all batteries were set to a capacity level of 10 mA*hour. The left half of figure 7a describes the remaining battery capacity of the transmitter after the file transfer, and the right half shows the receiver battery capacity. The solid bar in the figure indicates the total time for data transfer. Looking at the solid bar, it is clear that the sending the file at high data rate takes the least time thus the least battery capacity. Although the 2.4 kbps transmission and 19.2 kbps transmission took the same amount of time to transmit the data, 19.2 kbps pulse transmission saved much battery capacity due to the sleep period. Figure 7a also shows how different battery models exhibit different characteristics under different transmission methods. The linear model shows how optimum battery will behave as it shows the theoretical capacity of the battery under any discharge current. On the other hand, the rate dependent model accurately describes how real batteries will behave when there is a constant discharge for long duration. This is shown in 2.4 kbps transmission where the remaining capacity of rate dependent model is substantially less than the
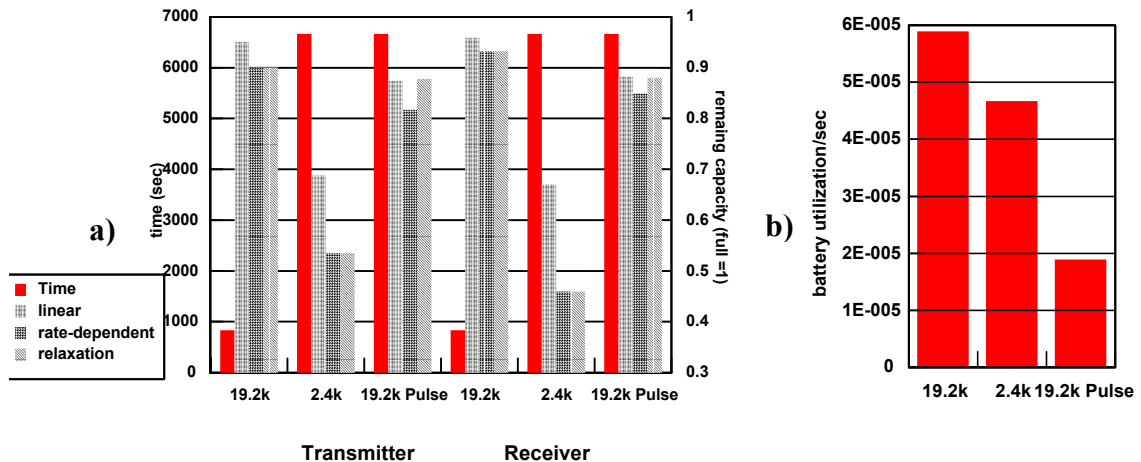
Figure 7: a) Battery Capacity Usage Under Different Discharge Profiles, b) Battery Utilization Rate

linear model. The other interesting model is the relaxation model which exhibits the both discharge rate dependent capacity and recovery effect. Since the relaxation model has recovery properties, the difference between relaxation model and rate dependent model is shown in the pulsed transmission cases. In figure 7a the relaxation model has the same remaining battery life as rate dependent model for 19.2kbps and 2.4 kbps continuous transmission and reception. However, in 19.2 kbps pulse transmission and reception, the relaxation model has almost equal capacity as the linear model due to the capacity recovery during the sleep mode.

5.2 Monitoring a Moving Vehicle in a Sensor Field

In this implementation we first show the effect of traffic on the sensing and communication traffic and then we evaluate simple power management scheme using the same sensor node setup as in the previous subsection. For this we have implemented a lightweight protocol stack similar to what one would expect to have on a tiny sensor node. The radio transmission and reception are driven by a TDMA based medium access control (MAC) protocol based on unique slot assignment algorithm derived from [9]. The MAC protocol assigns a unique slot to each node over a 2-hop radius and each node is aware of its one-hop neighbors and their corresponding slot assignment. For routing, we have implemented a very lightweight table-driven routing protocol with table size of one (next hop to user node). The motivation for TDMA scheme comes from our result in section 5.1 where a pulse transmission and reception can improve the battery utilization in the long term. In our power aware TDMA scheme this requirement is met for both the transmission and reception of packets. For transmission, a node is only allowed to transmit in its assigned time-slot. For reception, a node only needs to listen to the wireless channel for the duration of the slots that are already assigned to its one-hop neighbors. For the purposes of our discussion we refer to all the other remaining slots as *idle slots*.

In this scenario, a small cluster of 10 sensors equipped with seismic sensors is deployed to detect a bypassing truck as shown in figure 8a. The seismic sensors run at a sample rate of 400Hz to produce 16 bit samples. The sensor nodes are configured to report back to a gateway node that makes the results available on the Internet. Each sensor is programmed to transmit a report to the gateway within 5 seconds from the moment the ground vibrations from the truck are detected. If at least 2048 samples are obtained, the node can perform coherent detection and it will transmit a 10 byte to report the target type. This short packet is called "coherent traffic". If however, the node does not have enough samples at the end of the 5-second period, it enters a non-coherent detection mode and transmits all its available samples to the gateway node which performs sensor data combination (called "beamforming" [20]) to improve the detection accuracy. Since the sensor node transmits raw data when it enters non-coherent detection mode, the size of non-coherent data tends to be lot larger than the coherent traffic. This non coherent raw data is referred to as "non-coherent traffic" During the simulation, the network traffic will be consist of coherent and non-coherent traffic depending on whether the individual sensors successfully classified the target. We discuss the result of the simulation in the following two sub sections.

5.2.1 Efficiency of Power Management Scheme with TDMA

One apparent advantage of TDMA over other CSMA random access MAC protocols is the fact that the sensor nodes do not have to be in receive mode during the time slots where none of its neighbors are schedule to transmit. This allows the sensor nodes to perform a simple power management scheme that puts the CPU and radio to sleep during *idle slots* to conserve battery capacity. With this setup, we evaluate the efficiency of battery capacity utilization when this simple power management scheme is used. Figure 8a is the scenario used for our evaluation. It consists of 100 nodes uniformly distributed across a sensor field. The
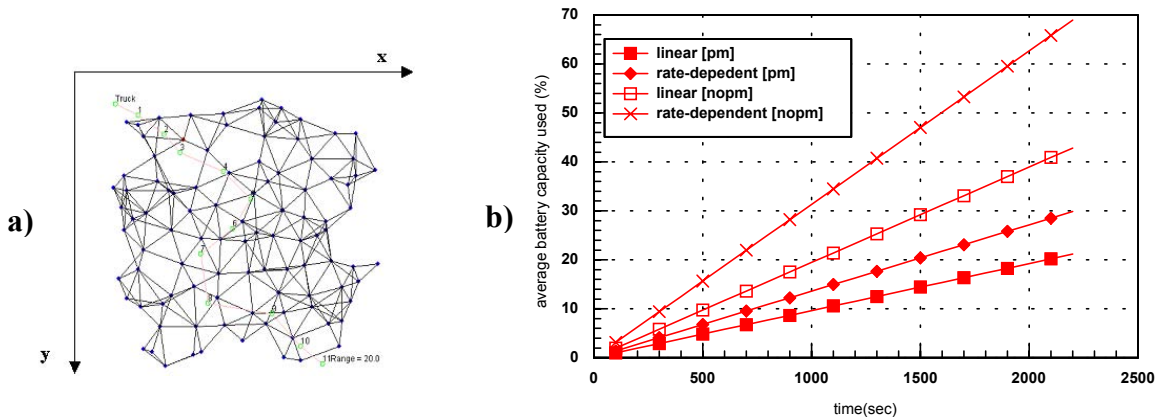
Figure 8: a) 100 Node Test Topology, b) Battery Capacity Usage

target travels at approximately 22 mph (10 m/s) through the track every 2 minutes. The target signals have an effective range of 20 meters. As the target travels through the sensor field, every node within the range of the target start collecting signal samples at 400 Hz, then send reports to the user node. We tested this scenario using the linear model and the rate dependent model by looking at battery utilization when the power management scheme is implemented (PM) and when there is no power management (NOPM).

The current drawn by each node will be similar to the cases described in section 5.1 with power management case resembling the 19.2 kbps pulse transmission and the no power management case resembling the 19.2 kbps continuous transmission. Figure 8b shows the average battery capacity utilization for each node. The bottom two curves show the difference in battery capacity utilization when the power management was used and the top two describe the cases when no power management is used. As the figure indicates, there is almost 100% improvement of battery utilization with the power management.

5.2.2 Effect of Sensor Power Cycle

In addition to the battery saving achieved by the TDMA power management scheme, we further look at how the sensor nodes can power cycle their sensors to conserve battery capacity. In this scenario (figure 9a), a square grid of sensor network is strategically placed over a flat field. The target travels along a pre-specified path and the sensor nodes attempts to make either coherent or non-coherent detection as described in the previous section. One difference in this scenario is that the sensor nodes attempt to turn on the sensors only intermittently to conserve power. When the sensor is turned off, the CPU of our experimental sensor node can go to mode 5 (table 1) where the power con-

sumption is in the range of microwatts. However, the trade-off comes from the reduction of detection and classification accuracy since the sensor will miss the sensor signals coming from the target when they are turned off. The cost of such missed events may be very application specific. If the target occurrence is very frequent, it may be okay to miss its detection, but if the occurrence is very infrequent, it may be very crucial to detect that one incidence. It is possible that the whole sensor network may have been deployed to detect that "one" incidence. Therefore, in designing sensor network it's crucial to look at the application requirement as well as the target characteristics to guarantee of certain quality of service (QoS) similar to the one provided in telecommunication network. One such QoS guarantee will be something like "a target with a 20 mph speed following this track will not pass through the sensor field undetected". In this section, we try to look at what would be the maximum battery power saving that can be achieved while providing such QoS guarantees.

We look at the impact of a simple power management scheme which randomly wakes up the sensor within a pre-specified time window of 100 seconds and stay up for different percentage of duration. Figure 9b shows the battery capacity used and the amount of coherent data bytes transmitted for different power cycle durations. The plot indicates that there is a rapid decrease in coherent detection as the power cycle percentage decrease from 60% to 50%. On the other hand, the battery utilization steadily decreases as the power cycle percentage decreases.

6 CONCLUSIONS

We have demonstrated a flexible toolset for studying power consumption in sensor networks. With the flexible architecture that closely simulate the behavior of real sensor network, accurate power models of sensor nodes and
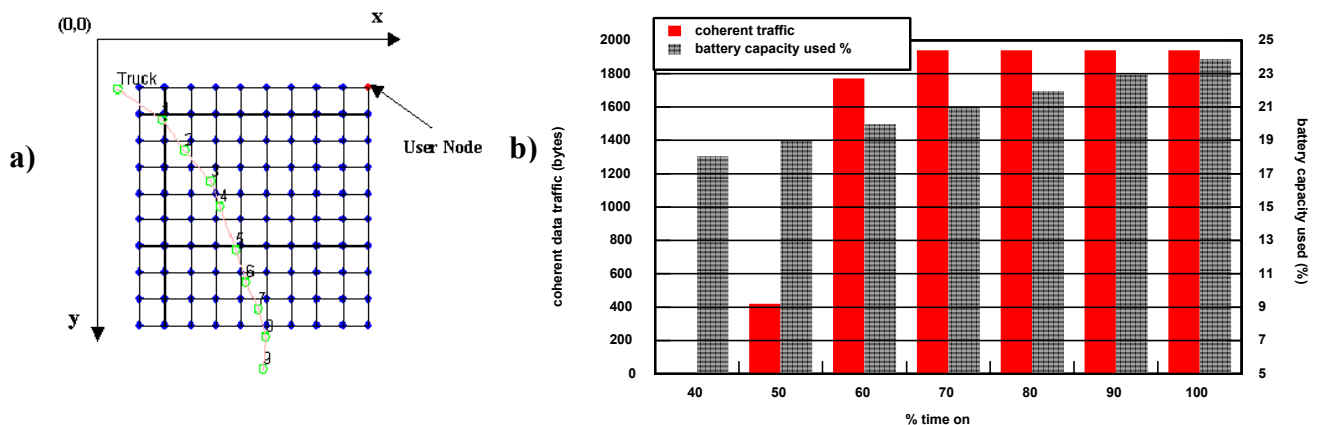
Figure 9: a) Sensor Scenario in Grid Sensor Network   b) Coherent Traffic Data and the Battery Efficiency Of Various Power Cycle Durations

analysis of battery behavior are utilized in a tool to evaluate power consumption in the context of a realistic scenario. With these results we can assess the power consumption for new sensor nodes that are currently under development. Furthermore, this tool has been an indispensable aid in estimating the resources required for the network protocols to function correctly in new node architectures. By simulating and validating target protocols we can also get a good indication of code size and memory requirements thus resulting in feasible low cost designs. We envision that this set of tools will play an instrumental role in the design and implementation of new application specific sensor networks.

REFERENCE

Chiasserini, C. F., and R. R. Rao. 1999. Pulsed battery discharge in communication devices. In *Proceedings of Mobicom 99*, Seattle, August 1999.

Fuller, T. F., M. Doyle, and J. Newman. 1994. Simulation and Optimization of the Dual Lithium Ion Insertion Cell. *Journal of Electrochem. Soc.*, vol. 141, no. 4, pp 1-10.

Linden, H.D. 1995. *Handbook of Batteries*. 2nd ed. New York: McGrawHill.

Matsushita Electric Corp. of America. 2001. *Panasonic lithium coin cell battery datasheet*. Available via `http://www.panasonic.com/industrial_ oem/battery/battery_oem/chem/lith/li th.htm` [accessed July 9, 2001].

ns-2 simulator. 2001. *ns-2 Simulator*, Available via `http://www.isi.edu/nsnam/ns/` [accessed July 9, 2001].

Rockwell Scientific Company LLC. 2001. *WINS (Wireless Integrated Network Systems) Project*. Available via `http://wins.rsc.rockwell.com/` [accessed July 9, 2001].

Simunic, T., L. Benini, and G. De Micheli. 1999. Energy-Efficient Design of Battery-Powered Embedded Systems. In *Proceedings of International Symposium on Low Power Electronics and Design*, 212-217, Piscataway, New Jersey.

Ulmer, C. 2001. *Sensor Network Simulator*, Available via `http://users.ece.gatech.edu/~grimace /research/sensorsimii/` [accessed July 9, 2001].

AUTHOR BIOGRAPHIES

SUNG PARK is a Ph.D. student of Electrical Engineering at University of California Los Angeles. He received his MS from Carnegie Mellon University in 1997. His research interests are network simulation and modeling, low power embedded systems, and sensor network. His email and web addresses are `<spark@ee.ucla.edu>` and `<http://www.ee.ucla.edu/~spark>`.

ANDREAS SAVVIDE is a Ph.D. student of Electrical Engineering at University of California Los Angeles. He received his MS from University of Massachusetts, Amherst in 1997. His research interests are adhoc wireless network, embedded system, and sensor network. His email and web addresses are `<asavvide@ee.ucla.edu>` and `<http://www.ee.ucla.edu/~asavvides>`.

MANI B. SRIVASTAVA is an Associate Professor of Electrical Engineering at University of California Los Angeles. He received his Ph.D. from U. C. Berkeley in 1992. His research interests are low power VLSI circuits, embedded systems, and sensor networks. His email and web addresses are `<mbs@ee.ucla.edu>` and `< http://nesl.ee.ucla.edu/people/mbs/>`.

Energy Efficient Wireless Scheduling: Adaptive Loading in Time

Curt Schurgers   Mani B. Srivastava

Networked and Embedded Systems Lab (NESL), Electrical Engineering Department,
University of California at Los Angeles (UCLA), CA 90095

Abstract— **When designing wireless systems, one of the major challenges is tackling the time depending fading behavior of the channel. To achieve maximum throughput under a power constraint, techniques have been proposed which adapt the modulation on the fly, based on the instantaneous channel condition. However, when the design goal is minimizing the overall energy consumption under a throughput constraint, we have to tackle a more complex scheduling problem. The reason is that energy optimality might require deliberately decreasing the transmission rate at times, if we know that the rate loss can be compensated for in the future when channel conditions are more favorable. We present a solution to the problem of minimum energy scheduling on wireless links by exploiting an analogy with the adaptive bit loding problem in multicarrier systems. Instead of allocating bits across multiple channels with different quality, we formulate the problem as one of allocating bits at the different time instants. The analogy is however not straightforward because one does not know the future, and therefore cannot simply apply conventional bit loading to the time dimension. We devise a new technique that approximates adaptive loading in time, but only depends on the instantaneous channel condition. Our algorithm is simple to implement, and shows upto 5x reduction in energy over existing approaches.**

I. INTRODUCTION

In wireless system, adapting the modulation at runtime has been proposed to maximize the throughput when faced with a variable channel [1]. Indeed, the non-constant fading behavior of the wireless medium results in attenuation variations, which can span orders of magnitude. When the channel is good, the modulation level is increased, while the opposite is done when the channel is bad. This principle of adaptive modulation has been explored extensively and can provide substantial throughput increase [2][3][4][5][6][7][8].

Systems are typically designed around the maximum expected traffic, such that they can handle the worst-case load. This is the operating premise of adaptive modulation: only by adapting to the channel conditions, can they reach the required throughput under a maximum power constraint. However, in typical wireless systems, the actual traffic load is often lower than the worst-case. One option is to transmit the data at the maximum rate, using modulation scaling, and afterwards shutdown the transceiver to save power.

In general, it has been shown that transmitting slower can actually further **reduce the energy consumption** of radio communications significantly [9]. This principle is analogous to voltage scaling in digital CMOS circuits, where voltage can be seen as a control knob to trade off energy and speed. However, the non-deterministically varying channel makes the problem different from voltage scaling.

In wireless communications, minimizing the energy by adapting the modulation has to take the channel variations into account, and becomes a problem of **wireless scheduling**. When the goal is to minimize the energy, and at the same time achieve a certain specified average throughput, we need to know the behavior of the channel in the future! The decision on the current rate critically depends on how good or bad the channel will be, *i.e.*, whether it is more efficient to send now or later. This issue does not arise when adapting the modulation to maximize the throughput, since only the current channel condition has to be taken into account there.

If we could magically know the channel over the entire time epoch, the problem reduces to one that has been studied extensively in multicarrier systems like OFDM (Orthogonal Frequency Division Multiplex) and DMT (Discrete Multi-Tone). In these systems, **adaptive bit loading** algorithms set the modulation level in each frequency band such that a predefined total number of bits are transmitted with minimum power, by leveraging the channel variations in the frequency domain. For our problem, we would simply need to replace the frequency dimension by the time dimension. However, as mentioned before, there is a severe practical problem with this approach: we cannot estimate the channel and then go back in time to set the modulation level. So, although adaptive bit loading in time would theoretically provide the best solution, it is impossible to use the existing techniques.

In this paper, we propose a new scheme that decides upon the modulation level **based solely on the current channel condition**. Our scheme, which we introduce in the next section, uses a set of thresholds that only depend on the channel statistics. Its performance gets arbitrarily close to that of ideal adaptive bit loading for increasing time windows, as we show in section III. The practical example of section IV illustrates the substantial energy gains that can be achieved.

II. THRESHOLD-BASED ALGORITHM

In the absence of a line-of-sight path between sender and receiver, narrowband wireless radio communications can be modeled as experiencing Rayleigh fading [10]. In this case, the normalized channel power-gain factor α has a probability density function (pdf) and cumulative distribution function (cdf) given by (1) and (2) respectively. These statistics are independent of the channel time-correlation, which is characterized by the Doppler rate.

$$f_\alpha(x) = e^{-x} \tag{1}$$

$$P(\alpha \le x) = F_\alpha(x) = 1 - e^{-x} \tag{2}$$

Equation (3) expresses the required transmit power P_i for Quadrature Amplitude Modulation (QAM) as a function of the modulation level b_i in bits per symbol and the channel gain factor α_i [10]. The results of this paper can be extended to other modulation schemes as well. The factor C is virtually constant over time and depends on the target error rate and the transmit distance, among others.

$$P_i = C \cdot \frac{\left(2^{b_i} - 1\right)}{\alpha_i} \qquad (3)$$

We refer to the transmit time instant of a symbol as a 'time sample'. In the remainder of this paper, the subindex i enumerates these time samples. In the case of multicarrier systems, the equivalent of a time sample would be the frequency band index.

Since there is no real bound on the number of time samples, we assume an infinite time duration in our analysis. In section III, we revisit the accuracy of this assumption. As a result, the intricacies of adaptive bit loading vanish, such that the task of selecting the correct modulation can be simplified considerably. In Appendix A, we prove this statement and derive a set of thresholds {d_1, d_2, d_3, etc.}, which uniquely couple each possible channel gain factor with its modulation level. This principle is illustrated in figure 1. The thresholds define non-overlapping ranges, each corresponding with a value of b. When the channel gain α is in a particular range, the modulation is set accordingly.

modulation level is kept constant. Our scheme is only valuable if the channel changes slowly enough. This is the same applicability range as other adaptive techniques, such as modulation adaptation [1][2][3][4][5][6][7].

In Appendix A, we prove that the thresholds are mutually related according to (4). Furthermore, threshold d_1 only depends on the pdf of the channel gains and the required average number of bits per symbol b_{av}. For Rayleigh fading, this relationship is given by (5). For other statistics, similar relationships can be found. We have essentially reduced the problem of knowing the exact behavior of the channel over time to that of finding its statistics.

$$d_{j+1} = d_1 \cdot 4^j \qquad (4)$$

$$b_{av} = 2 \cdot \sum_{j=0}^{\infty} \exp(-d_1 \cdot 4^j) \qquad (5)$$

The solid curves in figure 2 show the relationship of the thresholds and b_{av} as calculated from (4) and (5). To validate our derivations, we have also simulated what the threshold values would be when applying an existing loading algorithm for multicarrier systems on a large set of known independent time samples. The circles show to the lowest α that was encountered for each modulation level, which essentially corresponds to the threshold when the number of samples is very high. These simulations use the Hughes-Hartoghs loading algorithm [11], with the number of time samples N equal to 16384. We observe indeed a very tight correspondence between the simulated and theoretical values.

Figure 1: Relationship between thresholds and modulation
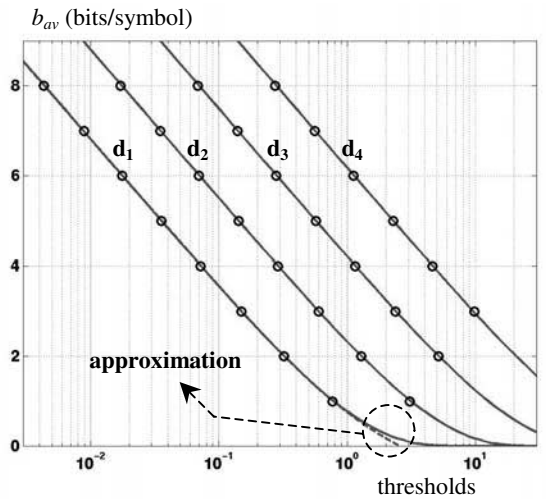
Figure 2: Ideal theoretical thresholds (solid lines) and simulation results for $N = 2^{14}$ (circles)

The knowledge of the channel gains is obtained through channel estimation, just as for adaptive modulation techniques that maximize the throughput [1]. To track variations, this estimation is updated regularly at a rate f_{update}, which is a compromise between overhead and performance degradations of inaccurate estimates. Between estimates, the

In practice, we want to know d_1 as a function of b_{av}, but it is hard to solve (4) analytically. The approximation of (6), obtained through curve fitting, is accurate down to $b_{av} = 1$. At

lower values, the bounds are underestimated, see figure 2, and more bits than needed are assigned. In this region, our algorithm is therefore slightly conservative.

$$\log_2(d_1) \approx 0.26 - 1.011 \cdot b_{av} + 1.115 \cdot 2^{-1.49 \cdot b_{av}} \qquad (6)$$

As explained before, we can map each α_i to its b_i by looking at how it relates to the thresholds. In Appendix B we show that the modulation level can be calculated as a direct function of the channel gain using (7), where negative values of b_i are saturated to zero. We can plug (6) directly into this equation, which introduces the dependency on the target average data rate.

$$b_i = 2 \cdot \left\lfloor 1 + \frac{1}{2} \cdot \log_2(\alpha_i) - \frac{1}{2} \cdot \log_2(d_1) \right\rfloor \qquad (7)$$

III. COMPARISON OF OUR ALGORITHM AND TRADITIONAL ADAPTIVE BIT LOADING

In this section, we evaluate how loading based on thresholds performs compared to traditional loading techniques. First we look at the achieved average data b_{pract}, which is defined as (8). N is the total number of time samples, which was made infinite in our derivation of the thresholds.

$$b_{pract} = \frac{1}{N} \cdot \sum_{i=1}^{N} b_i \qquad (8)$$

When we look at the average rate over a finite time window, the achieved b_{pract} might be different from the specified data rate that is used in (6). Figure 3 shows the pdf of b_{pract} for an example with $N = 1024$ uncorrelated Rayleigh fading time samples. We observe that the average rate is equal to b_{av}, but that there is some spread around this value. This data was collected in 10,000 simulation runs.

These simulations were repeated for different values of N and b_{av}. We observed that the mean value always corresponds to the target b_{av}. The variance is inversely proportional to N, where the proportionality factors, listed in Table I, only depend on b_{av}. This behavior is logical as the achieved data rate in (8) is the normalized sum of a large set of independent contributions and as such the central limit theorem applies.

TABLE I
NORMALIZED VARIANCE OF THE ACHIEVED RATE

b_{pract}	2	4	6	8
$\sigma^2_{b_{pract}} \cdot N$	2.27	3.22	3.57	3.60

As about 99.99% of all values lie within a band of 3.8 times the standard deviation on each side of the mean, we can virtually guarantee a deviation below 0.1 bits/symbol when $N > 5200$. If we allow this small jitter in data rate, our technique is useful in cases where we have at least that many independent time samples. Furthermore, when N goes to infinity, b_{pract} converges to b_{av}, which validates our derivation.

Since our ultimate goal is to reduce the energy consumption, we have also evaluated the total energy for the considered time window of N time samples. Figure 4 shows the resulting pdf for both our threshold-based approach and the Hughes-Hartoghs loading algorithm. It is interesting to note that although they have the same average behavior, our threshold-based scheme has a lower variance. The reason is that, in the case of a statistically worse channel, our threshold-based scheme would assign fewer bits. The traditional loading algorithms always exactly reach b_{av}, but these extra bits require the most additional power (which is proportional to the total energy).

Figure 3: Distribution of the achieved rate
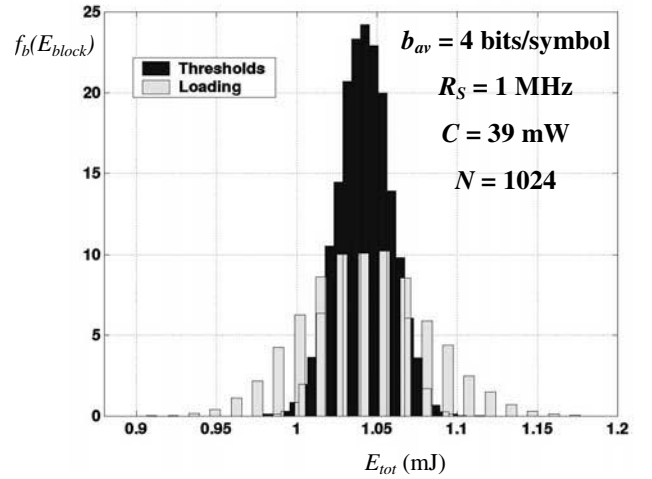
Figure 4: Distribution of the total energy

This same behavior can be observed for the average energy per bit E_{bit}, see figure 5. The mean value of relative difference between both techniques is equal to zero. The variance of this difference is listed in Table II, and as before, we encounter the same inverse proportionality with N due to the central limit theorem. When N increases, the variance will decrease to zero, and as a result, the behavior in terms of energy per bit of both schemes will become identical. In fact, for N going to infinity, our threshold-based algorithm results in the same solution as the traditional loading ones, such that all performance metrics are equal then.
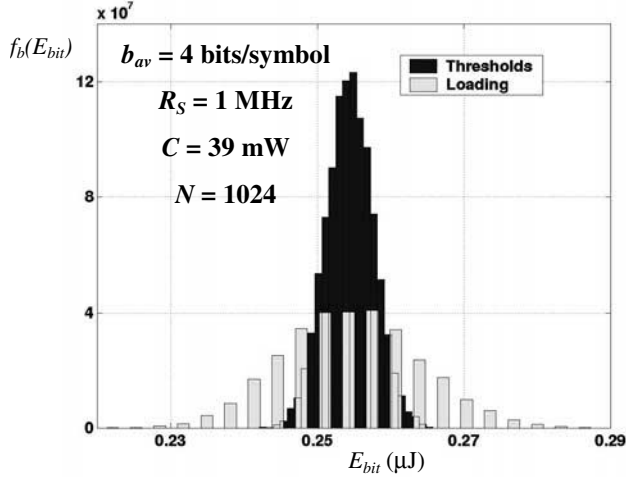
Figure 5: Distribution of the energy per bit

TABLE II

NORMALIZED VARIANCE OF THE RELATIVE DIFFERENCE IN ENERGY PER BIT

b_{pract}	2	4	6	8
$\sigma_{\partial E}^2 \cdot N$	0.80	0.96	1.12	1.20

IV. PERFORMANCE EVALUATION FOR A CORRELATED TIME FADING CHANNEL

In this section, we investigate the energy improvements for a practical wireless scheduling example. We choose a channel with correlated Rayleigh fading, corresponding to a Doppler rate of 50 Hz. Table III lists all relevant simulation parameters. The goal is to send a data file of 400 Mbit within 200 seconds, while minimizing the total energy consumption of this wireless transmission. We have the equivalent of about 10,000 ($= T_{tot} \cdot f_d$) independent time samples in this case. Besides the transmit power, given (3), we have also included the constant power P_{elec} of the electronics that perform all the other functionality in the radio, such as upconversion, frequency synthesis, etc. If the transmitter is shut down, no power is consumed.

TABLE III

SIMULATION PARAMETER SETTINGS

T_{tot}	200 s	C	39 mW
L_{tot}	400 Mbit	P_{elec}	180 mW
f_D	50 Hz	R_S	1 MHz
f_{update}	1 kHz	P_{max}	10 W

This system was designed to handle 6 bits/symbol as a maximum average throughput, but the current traffic load is less. Such situations are likely to occur in practical systems. A first option is to use traditional modulation adaptation, *i.e.* maximize the throughput, and shutdown the transceiver once the communication has finished. Our simulations show that in this case, the entire file is transmitted in 65.5 seconds, which corresponds to an average rate of 6.1 bits/symbol.

However, we could have transmitted the file at a constant rate of only 2 bits/symbol, such that the deadline would just be met. The effects of fading are counteracted by adjusting the transmit power. If this power is above the maximum limit P_{max}, no data is sent. As soon as the channel becomes better, we immediately compensate for this loss by temporarily raising the modulation level. This scheme therefore tries to operate with as flat a data rate as possible.

We have compared these two options with our wireless scheduling based on loading in time. The instantaneous modulation level is chosen according to (7). From table I, we work out that the standard deviation of b_{pract} is 0.018 bits/symbol, using $N = 10,000$. To compensate for the data rate jitter, we select b_{av} equal to 2.1 bits/symbol in (6) when calculating the thresholds. This corresponds to building in a time cushion of about 10 extra seconds. Our particular simulation transferred the file in 195.5 seconds.

TABLE IV

TOTAL ENERGY FOR THE FILE TRANSFER

Maximum throughput	Fixed throughput	Loading in time
E_{tot} = 341 J	E_{tot} = 133 J	E_{tot} = 71 J

* 0.4   *** 0.5**

Although all three options are able to complete the transfer within the specified deadline, their energy consumption vastly differs, as shown in Table IV. In this case, transmitting slower is superior to operating at maximum throughput and then shutting down. On top of the 2.5x reduction due to slowdown, loading cuts half the energy on top of that. Overall, the loading approach consumes about 5 times less than what would be achieved in a shutdown scenario.

V. CONCLUSIONS

We have presented a wireless scheduling approach that performs bit loading in time, based only on the instantaneous channel conditions when the statistics of the fading channel are known. It is able to provide the target throughput within a predictable error margin, while greatly reducing the energy consumption.

These energy gains essentially arise from the fact that for a particular average throughput, operating at a slower speed is superior to first transmitting at the highest speed and then shutting down. Strictly speaking, this statement is only valid when the electronics power does not dominate the transmit power. As this transmit power is a strong function of the communication distance, the conclusions presented here hold for typical systems, such a WLAN, cellular, terrestrial radio, etc. However, this situation might be different in ultra-short range communication setups, such as personal area networks [12] or sensor networks [13].

At least in principle, our algorithm can also be used for loading the frequency bands in wireless multicarrier systems. The benefit compared to existing techniques is an extremely simple implementation that only requires a one-shot algebraic operation on the channel gain factors. The disadvantage is the jitter in instantaneous data rate, as the number of independent samples is much less. For $N = 256$ independently fading subcarriers, the standard deviation would be around 0.1 bits/symbol, for example. Interestingly, the energy fluctuates less than with traditional loading algorithms.

VI. ACKNOWLEDGMENTS

APPENDIX A

In this appendix, we derive the analytical relationships that link the thresholds to the target data rate. Since we are considering an infinite number of time samples, the pdf completely describes the channel behavior, as their correlation becomes irrelevant. We have omitted the time subindex i for readability reasons. Expression (3) for transmit power is repeated here as (9). The modulation level is typically an even integer and can thus be written as (10) [10].

$$P = C \cdot \frac{\left(2^b - 1\right)}{\alpha} \qquad (9)$$

$$b = 2 \cdot j \qquad (10)$$

When the modulation is raised from *(2j)* to *(2j+2)*, the increase in power is:

$$\Delta P^{j+1} = P^{j+1} - P^j = C \cdot \frac{\left(2^{2 \cdot j + 2} - 2^{2 \cdot j}\right)}{\alpha} = 3 \cdot C \cdot \frac{4^j}{\alpha} \qquad (11)$$

The adaptive loading algorithm [11] increases the modulation level on the samples that require the least extra power, until the target date rate is reached. This way, the solution is optimal in terms of power. We aggregate the samples that have the same j in groups, which are consequently identified by their value j. Since P is inversely proportional to α, the samples in group *j+1* all have a value of α that is larger than that of all the samples in group j. As a result, these groups appear as non-overlapping regions on the cdf, see figure 6.

In each group, the sample that has the lowest ΔP, which we call the representative sample for group j, is the first one to receive a higher modulation and therefore switch groups. This is also the sample that has the highest value of α in the group. Since we are considering an infinite number of samples, the ΔP of all the representative samples is always equal, or equivalently:

$$\Delta P^{j+1} = \Delta P^{k+1} \qquad \forall j, k \in \aleph \qquad (12)$$

In figure 6, we see that threshold d_{j+1} corresponds to the channel gain α of the representative sample of group j. We can derive the relationship between the thresholds from (11) and (12):

$$3 \cdot C \cdot \frac{4^j}{d_{j+1}} = 3 \cdot C \cdot \frac{4^k}{d_{k+1}} \qquad (13)$$

$$\Downarrow$$

$$d_{j+1} = d_{k+1} \cdot 4^{j-k} = d_j \cdot 4 \qquad (14)$$

Or equivalently, we can write the thresholds as a function of d_1, as (15).

$$d_{j+1} = d_1 \cdot 4^j \qquad (15)$$

Figure 6 also illustrates that we can write the average data rate as the sum of the data rate in each region, multiplied by the probability of being in that region:

$$b_{av} = \sum_{j=1}^{\infty} (2 \cdot j) \cdot \left(F(d_{j+1}) - F(d_j)\right) \qquad (16)$$

Together with (2), this equation can be rewritten as (17). By substituting (15), we arrive at the final expression (18) that links the average throughput and threshold d_1.

$$b_{av} = 2 \cdot \sum_{j=1}^{\infty} j \cdot \left(e^{-d_j} - e^{-d_{j+1}} \right) = 2 \cdot \sum_{j=1}^{\infty} e^{-d_j} \qquad (17)$$

$$b_{av} = 2 \cdot \sum_{j=0}^{\infty} \exp(-d_1 \cdot 4^j) \qquad (18)$$

APPENDIX B

In this appendix, we derive the explicit relationship between the channel gain and modulation level. By reordering the terms of (15), and taking into account (10), we can express the value of b at each threshold as:

$$b = 2 \cdot j = \log_2\left(\frac{d_{j+1}}{d_1}\right) + 2 \qquad (19)$$

Every encountered channel gain α_i is mapped to the next lower threshold. After some algebraic manipulations, we can express this operation as (20), where $\lfloor . \rfloor^*$ denotes rounding to the next lower integer and saturating negative values to zero.

$$b = 2 \cdot \left\lfloor \frac{1}{2} \cdot \log_2\left(\frac{\alpha}{d_1}\right) + 1 \right\rfloor^* \qquad (20)$$
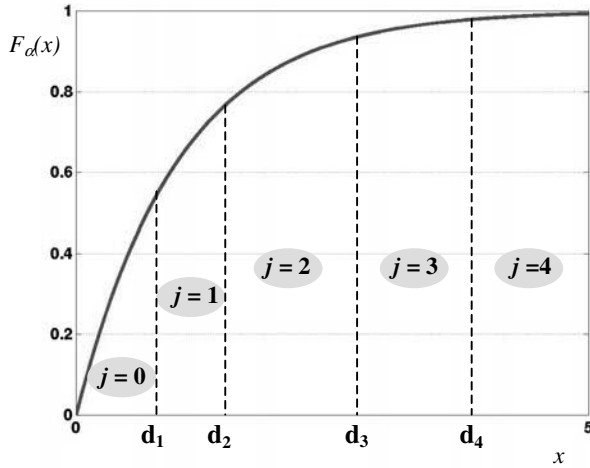
Figure 6: Relationship between thresholds and cdf

REFERENCES

[1] Jacobsmeyer, J., "Adaptive data rate communications for high frequency radio channels," *MILCOM'91*, McLean, VA, pp. 938-942, 1991.

[2] Webb, W., Steele, R., "Variable rate QAM for mobile radio," *IEEE Trans. on Comm.*, Vol.43, No.7, pp. 2223-2230, July 1995.

[3] Hamaguchi, K., Kamio, Y., Moriyama, E., "Implementation and performance of an adaptive QAM modulation-level-controlled system for land mobile communications," *VTC'97*, Phoenix, AZ, pp. 1214-1218, May 1997.

[4] Goldsmith, A.J., Soon-Ghee Chua, "Variable-rate variable-power MQAM for fading channels," *IEEE Trans. on Comm.*, Vol.45, No.10, pp. 1218-1230, Oct. 1997.

[5] Zhao, Y., "Theoretical study of link adaptation algorithms for adaptive modulation in wireless mobile communication systems," *ICUPC'98*, Florence, Italy, pp. 587-591, Oct. 1998.

[6] Ue, T., Sampei, S., Morinaga, N., Hamaguchi, K., "Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD system for high-bit rate wireless data transmission," *IEEE Trans. on Vehicular Technology*, Vol.47, No.4, pp. 1134-1147, Nov. 1998.

[7] Balachandran, K., Kadaba, S., Nanda, S., "Channel quality estimation and rate adaptation for cellular mobile radio," *IEEE JSAC*, Vol.17, No.7, pp. 1244-1256, July 1999.

[8] Kyung-Ho Cho, Samueli, H., "A frequency-agile single-chip QAM modulator with beamforming diversity," *IEEE Journal of Solid-State Circuits*, Vol.36, No.3, pp. 398-407, March 2001.

[9] Schurgers, C., Aberthorne, O., Srivastava, M., "Modulation scaling for energy aware communication systems," *ISLPED'01*, Huntington Beach, CA, pp. 96-99, Aug. 2001.

[10] Proakis, J., "Digital Communications," *McGraw-Hill Series in Electrical and Computer Engineering*, 3rd Edition, 1995.

[11] Hughes-Hartogs, D., "Ensemble modem structure for imperfect transmission media," *U.S. Patents*, No. 4,679,227 (July 1987), 4,731,816 (March 1988), 4,833,796 (May 1989).

[12] Siep, T., Gifford, I., Braley, R., Heile, R., "Paving the way for personal area network standards: an overview of the IEEE P802.15 Working Group for Wireless Personal Area Networks," *IEEE Personal Communications*, Vol.7, No.1, pp.37-43, Feb. 2000.

[13] Sohrabi, K., Gao, J., Ailawadhi, V., Pottie, G., "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications Magazine*, Vol.7, No.5, pp. 16-27, Oct. 2000.

Energy Aware Wireless Sensor Networks

Vijay Raghunathan, Curt Schurgers, Sung Park, and Mani B. Srivastava
Department of Electrical Engineering, University of California, Los Angeles, CA 90095.

INTRODUCTION

Self-configuring wireless sensor networks can be invaluable in many civil and military applications for collecting, processing, and disseminating wide ranges of complex environmental data. They have therefore, attracted considerable research attention in the last few years. The WINS [1] and SmartDust [2] projects for instance, aim to integrate sensing, computing, and wireless communication capabilities into a small form factor to enable low-cost production of these tiny nodes in large numbers. Several other groups are investigating efficient hardware/software system architectures, signal processing algorithms, and network protocols for wireless sensor networks [3], [4], [5].

Sensor nodes are battery-driven, and hence operate on an extremely frugal energy budget. Further, they must have a lifetime on the order of months to years, since battery replacement is not an option for networks with thousands of physically embedded nodes. In some cases, these networks may be required to operate solely on energy scavenged from the environment through seismic, photovoltaic, or thermal conversion. This transforms energy consumption into the most important factor that determines sensor node lifetime.

Conventional low-power design techniques [6] and hardware architectures only provide point solutions which are insufficient for these highly energy constrained systems. Energy optimization, in the case of sensor networks, is much more complex, since it involves not only reducing the energy consumption of a single sensor node, but also maximizing the lifetime of an entire network. The network lifetime can be maximized only by incorporating energy-awareness into every stage of wireless sensor network design and operation, thus empowering the system with the ability to make dynamic tradeoffs between energy consumption, system performance, and operational fidelity. This new networking paradigm, with its extreme focus on energy efficiency, poses several system and network design challenges that need to be overcome to fully realize the potential of the wireless sensor systems.

A quite representative application in wireless sensor networks is event tracking, which has widespread use in applications such as security surveillance and wildlife habitat monitoring. Tracking involves a significant amount of collaboration between individual sensors to perform complex signal processing algorithms such as Kalman Filtering, Bayesian Data Fusion, and Coherent Beamforming. This collaborative signal processing nature of sensor networks offers significant opportunities for energy management. For example, just the decision of whether to do the collaborative signal processing at the user end-point or somewhere inside the network has significant implication on energy and lifetime. We will use tracking as the driver to illustrate many of the techniques presented in this paper.

PAPER OVERVIEW

This paper describes architectural and algorithmic approaches that designers can use to enhance the energy awareness of wireless sensor networks. The paper starts off with an analysis of the power consumption characteristics of typical sensor node architectures, and identifies the various factors that affect system lifetime. We then present a suite of techniques that perform aggressive energy optimization while targeting all stages of sensor network design, from individual nodes to the entire network. Maximizing network lifetime requires the use of a well-structured design methodology, which enables energy aware design, and operation of all aspects of the sensor network, from the underlying hardware platform, to the application software and network protocols. Adopting such a holistic approach ensures that energy awareness is incorporated not only into individual sensor nodes, but also into groups of communicating nodes, and the entire sensor network. By following an energy-aware design methodology based on techniques such as in this paper, designers can enhance network lifetime by orders of magnitude.

WHERE DOES THE POWER GO?

The first step in designing energy aware sensor systems involves analyzing the power dissipation characteristics of a wireless sensor node. Systematic power analysis of a sensor node is extremely important to identify power bottlenecks in the system, which can then be the target of aggressive optimization. We analyze two popular sensor nodes from a power consumption perspective, and discuss how decisions taken during node design can significantly impact the system energy consumption.

The system architecture of a canonical wireless sensor node is shown in Figure 1. The node is comprised of four subsystems: (i) a computing subsystem consisting of a microprocessor or microcontroller, (ii) a communication subsystem consisting of a short range radio for wireless communication, (iii) a sensing subsystem that links the node to the physical world and

consists of a group of sensors and actuators, and (iv) a power supply subsystem, which houses the battery and the DC-DC converter, and powers the rest of the node. The sensor node shown in Figure 1 is representative of commonly used node architectures such as [1], [2].

MICRO CONTROLLER UNIT (MCU)

Providing intelligence to the sensor node, the MCU is responsible for control of the sensors, and execution of communication protocols and signal processing algorithms on the gathered sensor data. Commonly used MCUs are Intel's StrongARM microprocessor and Atmel's AVR microcontroller. The power-performance characteristics of MCUs have been studied extensively, and several techniques have been proposed to estimate the power consumption of these embedded processors [7], [8]. While the choice of MCU is dictated by the required performance levels, it can also significantly impact the node's power dissipation characteristics. For example, the StrongARM microprocessor from Intel, used in high end sensor nodes, consumes around 400mW of power while executing instructions, whereas the ATmega103L AVR microcontroller from Atmel consumes only around 16.5mW, but provides much lower performance. Thus, the choice of MCU should be dictated by the application scenario, to achieve a close match between the performance level offered by the MCU, and that demanded by the application. Further, MCUs usually support various operating modes, including *Active, Idle*, and *Sleep* modes, for power management purposes. Each mode is characterized by a different amount of power consumption. For example, the StrongARM consumes 50mW of power in the *Idle* mode, and just 0.16mW in the *Sleep* mode. However, transitioning between operating modes involves a power and latency overhead. Thus, the power consumption levels of the various modes, the transition costs, and the amount of time spent by the MCU in each mode, all have a significant bearing on the total energy consumption (battery lifetime) of the sensor node.

RADIO

The sensor node's radio enables wireless communication with neighboring nodes and the outside world. There are several factors that affect the power consumption characteristics of a radio, including the type of modulation scheme used, data rate, transmit power (determined by the transmission distance), and the operational duty cycle. In general, radios can operate in four distinct modes of operation, namely *Transmit, Receive, Idle,* and *Sleep* modes. An important observation in the case of most radios is that, operating in *Idle* mode results in significantly high power consumption, almost equal to the power consumed in the

Receive mode [11]. Thus, it is important to completely shutdown the radio rather than transitioning to *Idle* mode, when it is not transmitting or receiving data. Another influencing factor is that, as the radio's operating mode changes, the transient activity in the radio electronics causes a significant amount of power dissipation. For example, when the radio switches from sleep mode to transmit mode to send a packet, a significant amount of power is consumed for starting up the transmitter itself [9].

SENSORS

Sensor transducers translate physical phenomena to electrical signals, and can be classified as either analog or digital devices depending on the type of output they produce. There exist a diversity of sensors that measure environmental parameters such as temperature, light intensity, sound, magnetic fields, image *etc*. There are several sources of power consumption in a sensor, including (i) signal sampling and conversion of physical signals to electrical ones, (ii) signal conditioning, and (iii) analog to digital conversion. Given the diversity of sensors there is no typical power consumption number. In general, however, passive sensors such as temperature, seismic *etc*., consume negligible power relative to other components of sensor node. However, active sensors such as sonar rangers, array sensors such as imagers, and narrow field-of-view sensors that require repositioning such as cameras with pan-zoom-tilt can be large consumers of power.

POWER ANALYSIS OF SENSOR NODES

Table I shows the power consumption characteristics of Rockwell's WINS node [10], which represents a high-end sensor node, and is equipped with a powerful StrongARM SA-1100 processor from Intel, a radio module from Conexant Systems, and several sensors including acoustic and seismic ones. Table II gives the characteristics of the MEDUSA-II, an experimental sensor node developed at the Networked and Embedded Systems Lab, UCLA. The MEDUSA node, designed to be ultra low power, is a low-end sensor node similar to the COTS Motes developed as part of the SmartDust project [2]. It is equipped with an AVR microcontroller from ATMEL, a low-end RFM radio module, and a few sensors. As can be seen from the tables, the power dissipation characteristics of the two nodes differ significantly. There are several inferences that can be drawn from these tables:

Using low-power components and trading off unnecessary performance for power savings during node design, can have a significant impact, up to a few orders of magnitude.

The node power consumption is strongly dependent on the operating modes of the components. For example, as Table I shows, the WINS node consumes only around one sixth the power when the MCU is in *Sleep* mode, than when it is in *Active* mode.

Due to extremely small transmission distances, the power consumed while receiving data can often be greater than the power consumed while transmitting packets, as is evident from Figure 2. Thus, conventional network protocols which usually assume the receive power to be negligible, are no longer efficient for sensor networks, and customized protocols which explicitly account for receive power have to be developed instead.

The power consumed by the node with the radio in *Idle* mode is approximately the same with the radio in *Receive* mode. Thus, operating the radio in *Idle* mode does not provide any advantage in terms of power. Previously proposed network protocols have often ignored this fact, leading to fallacious savings in power consumption, as pointed out in [11]. Therefore, the radio should be completely shut off whenever possible, to obtain energy savings.

BATTERY ISSUES

The battery supplies power to the complete sensor node, and hence plays a vital role in determining sensor node lifetime. Batteries are complex devices whose operation depends on many factors including battery dimensions, type of electrode material used, and diffusion rate of the active materials in the electrolyte. In addition, there can be several non-idealities that can creep in during battery operation, which adversely affect system lifetime. We describe the various battery non-idealities, and discuss system level design approaches that can be used to prolong battery lifetime.

Rated capacity effect

The most important factor that affects battery lifetime is the discharge rate or the amount of current drawn from the battery. Every battery has a rated current capacity, specified by the manufacturer. Drawing higher current than the rated value leads to a significant reduction in battery life. This is because, if a high current is drawn from the battery, the rate at which active ingredients diffuse through the electrolyte falls behind the rate at which they are consumed at the electrodes. If the high discharge rate is maintained for a long time, the electrodes run out of active materials, resulting in battery death even though active ingredients are still present in the electrolyte. Hence, to avoid battery life degradation, the amount of current drawn from the battery should be kept under tight check. Unfortunately, depending on the

battery type (Lithium Ion, NiMH, NiCd, Alkaline, *etc.*), the minimum required current consumption of sensor nodes often exceeds the rated current capacity, leading to sub-optimal battery lifetime.

Relaxation effect

The effect of high discharge rates can be mitigated to a certain extent through battery relaxation. If the discharge current from the battery is cut off or reduced, the diffusion and transport rate of active materials catches up with the depletion caused by the discharge. This phenomenon is called the relaxation effect, and enables the battery to recover a portion of its lost capacity. Battery lifetime can be significantly increased if the system is operated such that the current drawn from the battery is frequently reduced to very low values, or is completely shut off [12].

DC-DC CONVERTER

The DC-DC converter is responsible for providing a constant supply voltage to the rest of the sensor node while utilizing the complete capacity of the battery. The efficiency factor associated with the converter plays a big role in determining battery lifetime [13]. A low efficiency factor leads to significant energy loss in the converter, reducing the amount of energy available to other sensor node components. Also, the voltage level across the battery terminals constantly decreases as it gets discharged. The converter therefore draws increasing amounts of current from the battery to maintain a constant supply voltage to the sensor node. As a result, the current drawn from the battery becomes progressively higher than the current that actually gets supplied to the rest of the sensor node. This leads to depletion in battery life due to the rated capacity effect, as explained earlier. Figure 3 shows the difference in current drawn from the battery and the current delivered to the sensor node for a Lithium-Ion coin cell battery.

NODE LEVEL ENERGY OPTIMIZATION

Having studied the power dissipation characteristics of wireless sensor nodes, we now focus our attention to the issue of minimizing the power consumed by these nodes. As a first step towards incorporating energy awareness into the network, it is necessary to develop hardware/software design methodologies and system architectures that enable energy-aware design and operation of individual sensor nodes in the network.

POWER-AWARE COMPUTING

Advances in low-power circuit and system design [6] have resulted in the development of several ultra low

power microprocessors, and microcontrollers. In addition to using low-power hardware components during sensor node design, operating the various system resources in a power-aware manner through the use of dynamic power management (DPM) [14] can reduce energy consumption further, increasing battery lifetime. A commonly used power management scheme is based on idle component shutdown, in which the sensor node, or parts of it, is shutdown or sent into one of several low-power states if no interesting events occur. Such event-driven power management is extremely crucial in maximizing node lifetime. The core issue in shutdown based DPM is deciding the state transition policy [14], since different states are characterized by different amounts of power consumption, and state transitions have a non-negligible power and time overhead.

While shutdown techniques save energy by turning off idle components, additional energy savings are possible in active state through the use of dynamic voltage scaling (DVS) [15]. Most microprocessor-based systems have a time varying computational load, and hence peak system performance is not always required. DVS exploits this fact by dynamically adapting the processor's supply voltage and operating frequency to just meet the instantaneous processing requirement, thus trading off unutilized performance for energy savings. DVS based power management, when applicable, has been shown to have significantly higher energy efficiency compared to shutdown based power management due to the convex nature of the energy- speed curve [15]. Several modern processors such as Intel's StrongARM and Transmeta's Crusoe support scaling of voltage and frequency, thus providing control knobs for energy-performance management.

For example, consider the target-tracking application discussed earlier. The duration of node shutdown can be used as a control knob to trade off tracking fidelity against energy. A low operational duty cycle for a node reduces energy consumption at the cost of a few missed detections. Further, the target update rate varies, depending on the Quality of Service requirements of the user. A low update rate implies more available latency to process each sensor data sample, which can be exploited to reduce energy through the use of DVS.

ENERGY AWARE SOFTWARE

Despite the higher energy efficiency of application specific hardware platforms, the advantage of flexibility offered by microprocessor and DSP based systems has resulted in the increasing use of programmable solutions during system design. Sensor network lifetime can be significantly enhanced if the system software, including the operating system (OS), application layer, and network protocols are all designed to be energy aware.

The OS is ideally poised to implement shutdown-based and DVS-based power management policies, since it has global knowledge of the performance and fidelity requirements of all the applications, and can directly control the underlying hardware resources, fine tuning the available performance-energy control knobs. At the core of the OS is a task scheduler, which is responsible for scheduling a given set of tasks to run on the system while ensuring that timing constraints are satisfied. System lifetime can be increased considerably by incorporating energy awareness into the task scheduling process [16], [17].

The energy aware real-time scheduling algorithm proposed in [16] exploits two observations about the operating scenario of wireless systems, to provide an adaptive power *vs*. fidelity tradeoff. The first observation is that these systems are inherently designed to operate resiliently in the presence of varying fidelity in the form of data losses, and errors over wireless links. This ability to adapt to changing fidelity is used to trade off against energy. Second, these systems exhibit significant correlated variations in computation and communication processing load due to underlying time-varying physical phenomena. This observation is exploited to proactively manage energy resources by predicting processing requirements. The voltage is set according to predicted computation requirements of individual task instances, and adaptive feedback control is used to keep the system fidelity (timing violations) within specifications.

The energy-fidelity tradeoff can be exploited further by designing the application layer to be energy scalable. This can be achieved by transforming application software such that the most significant computations are performed first. Thus, terminating the algorithm prematurely due to energy constraints, does not impact the result severely. For example, the target tracking application described earlier involves the extensive use of signal filtering algorithms such as Kalman filtering. Transforming the filtering algorithms to be energy scalable, trades off computational precision (and hence, tracking precision) for energy consumption. Several transforms to enhance the energy scalability of DSP algorithms are presented in [18].

POWER MANAGEMENT OF RADIOS

While power management of embedded processors has been studied extensively, incorporating power awareness into radio subsystems has remained relatively unexplored. Power management of radios is extremely important since wireless communication is a major power consumer during system operation. One way of characterizing the importance of this problem is in terms of the ratio of the energy spent in sending one bit to the energy spent in executing one instruction. While it is not

quite fair to compare this ratio across nodes without normalizing for transmission range, bit error probability, and the complexity of instruction (8-bit vs. 32-bit), this ratio is nevertheless useful. Example values are from 1500 to 2700 for Rockwell's WIN nodes, 220 to 2900 for the MEDUSA II nodes, and is around 1400 for the WINS NG 2.0 nodes from the Sensoria Corporation that are used by many researchers.

The power consumed by a radio has two main components to it, an RF component that depends on the transmission distance and modulation parameters, and an electronics component that accounts for the power consumed by the circuitry that performs frequency synthesis, filtering, up-converting, *etc*. Radio power management is a non-trivial problem, particularly since the well-understood techniques of processor power management may not be directly applicable. For example, techniques such as dynamic voltage and frequency scaling reduce processor energy consumption at the cost of an increase in the latency of computation. However, in the case of radios, the electronics power can be comparable to the RF component (which varies with the transmission distance). Therefore, slowing down the radio may actually lead to an increase in energy consumption. Other architecture specific overheads like the startup cost of the radio can be quite significant [9], making power management of radios a complex problem. The various tradeoffs involved in incorporating energy awareness into wireless communication will be discussed further in the next section.

ENERGY AWARE PACKET FORWARDING

In addition to sensing and communicating its own data to other nodes, a sensor node also acts as a router, forwarding packets meant for other nodes. In fact, for typical sensor network scenarios, a large portion (around 65%) of all packets received by a sensor node need to be forwarded to other destinations [19]. Typical sensor node architectures implement most of the protocol processing functionality on the main computing engine. Hence, every received packet, irrespective of its final destination, travels all the way to the computing subsystem and gets processed, resulting in a high energy overhead. The use of intelligent radio hardware, as shown in Figure 4, enables packets that need to be forwarded to be identified and re-directed from the communication subsystem itself, allowing the computing subsystem to remain in *Sleep* mode, saving energy [19].

ENERGY AWARE WIRELESS COMMUNICATION

While power management of individual sensor nodes reduces energy consumption, it is important for the communication between nodes to be conducted in an energy efficient manner as well. Since the wireless transmission of data accounts for a major portion of the total energy consumption, power management decisions that take into account the effect of inter-node communication yield significantly higher energy savings. Further, incorporating power management into the communication process enables the diffusion of energy awareness from an individual sensor node to a group of communicating nodes, thereby enhancing the lifetime of entire regions of the network. To achieve power-aware communication it is necessary to identify and exploit the various performance-energy trade-off knobs that exist in the communication subsystem.

MODULATION SCHEMES

Besides the hardware architecture itself, the specific radio technology used in the wireless link between sensor nodes plays an important role in energy considerations. The choice of modulation scheme greatly influences the overall energy versus fidelity and latency tradeoff that is inherent to a wireless communication link. Equation (1) expresses the energy cost for transmitting one bit of information, as a function of the packet payload size L, the header size H, the fixed overhead E_{start} associated with the radio startup transient, and the symbol rate R_S for an M-ary modulation scheme [9], [20]. P_{elec} represents the power consumption of the electronic circuitry for frequency synthesis, filtering, modulating, upconverting, *etc*. The power delivered by the power amplifier, P_{RF}, needs to go up as M increases, in order to maintain the same error rate.

$$Ebit = \frac{Estart}{L} + \frac{Pelec + PRF(M)}{Rs * \log_2 M} * (1 + \frac{H}{L}) \quad (1)$$

Figure 5 plots the communication energy per bit as a function of the packet size and the modulation level M. This curve was obtained using the parameters given in Table III, which are representative for sensor networks, and choosing Quadrature Amplitude Modulation (QAM) [9], [20]. The markers in Figure 5 indicate the optimal modulation setting for each packet size, which is independent of L. In fact, this optimal modulation level is relatively high, close to 16-QAM for the values specified in Table III. Higher modulation levels might be unrealistic in low-end wireless systems, such as sensor nodes. In these scenarios, a practical guideline for saving energy is to transmit as fast as possible, at the optimal setting [9]. However, if for reasons of peak-throughput, higher modulation levels than the optimal one need to be provided, adaptively changing the modulation level can lower the overall energy consumption. When the

instantaneous traffic load is lower than the peak value, transmissions can be slowed down, possibly all the way to the optimal operating point. This technique of dynamically adapting the modulation level to match the instantaneous traffic load, as part of the radio power management, is called modulation scaling [20]. It is worth noting that dynamic modulation scaling is the exact counterpart of dynamic voltage scaling, which has been shown to be extremely effective for processor power management, as described earlier.

The above conclusions are expected to hold for other implementations of sensor network transceivers as well. Furthermore, since the startup cost is significant in most radio architectures [9], it is beneficial to operate with as large a packet size as possible, since it amortizes this fixed overhead over more bits. However, aggregating more data into a single packet has the downside of increasing the overall latency of information exchange.

The discussion up until now has focused on the links between two sensor nodes, which are characterized by their short distance. However, when external users interact with the network, they often times do so via specialized gateway nodes [22], [23]. These gateway nodes offer long-haul communication services, and are therefore in a different regime where P_{RF} dominates P_{elec}. In this case, the optimal M shifts to the lowest possible value, such that it becomes beneficial to transmit as slow as possible, subject to the traffic load. In this regime, modulation scaling is clearly very effective [20].

COORDINATED POWER MANAGEMENT TO EXPLOIT COMPUTATION COMMUNICATION TRADEOFF

Sensor networks involve several node-level and network-wide computation-communication tradeoffs, which can be exploited for energy management. At the individual node level, power management techniques such as DVS and modulation scaling reduce the energy consumption at the cost of increased latency. Since both the computation and communication subsystems take from the total acceptable latency budget, exploiting the inherent synergy between them to perform coordinated power management will result in far lower energy consumption. For example, the relative split up of the available latency for the purposes of dynamic voltage scaling and dynamic modulation scaling significantly impacts the energy savings obtained. Figure 6 shows a system power management module that is integrated into the OS, and performs coordinated power management of the computing, communication and sensing subsystems.

The computation-communication tradeoff manifests itself in a powerful way due to the distributed nature of these sensor networks. The network's inherent capability for parallel processing offers further energy optimization potential. Distributing an algorithm's computation among multiple sensor nodes enables the computation to be performed in parallel. The increased allowable latency per computation enables the use of voltage scaling, or other energy-latency tradeoff techniques. Distributed computing algorithms however demand more inter-node collaboration, thereby increasing the amount of communication that needs to take place.

These computation-communication tradeoffs extend beyond individual nodes to the network level too. As we will discuss in the next section, the high redundancy present in the data gathering process, enables the use of data combining techniques to reduce the amount of data to be communicated, at the expense of extra computation at individual nodes to perform data aggregation.

LINK LAYER OPTIMIZATIONS

While exploring energy-performance-quality tradeoffs, reliability constraints also have to be considered, which are related to the interplay of communication packet losses and sensor data compression. Reliability decisions are usually taken at the link layer, which is responsible for some form of error detection and correction. Adaptive error correction schemes were proposed in [24] to reduce energy consumption, while maintaining the bit error rate (BER) specifications of the user. For a given BER requirement, error control schemes reduce the transmit power required to send a packet, at the cost of additional processing power at the transmitter and receiver. This is especially useful for long-distance transmissions to gateway nodes, which involve large transmit power. Link layer techniques also play an indirect role in reducing energy consumption. The use of a good error control scheme minimizes the number of times a packet retransmissions, thus reducing the power consumed at the transmitter as well as the receiver.

NETWORK-WIDE ENERGY OPTIMIZATION

Incorporating energy awareness into individual nodes and pairs of communicating nodes alone does not solve the energy problem in sensor networks. The network as a whole should be energy-aware, for which the network-level global decisions should be energy-aware.

TRAFFIC DISTRIBUTION

At the highest level of sensor network, the issue of how traffic is forwarded from the data source to the data sink arises. Data sinks typically are user nodes or specialized gateways that connect the sensor network to the outside world. One aspect of traffic forwarding is the choice of an energy efficient multi-hop route between source and destination. Several approaches have been

proposed [3], [23], [25] which aim at selecting a path that minimizes the total energy consumption.

However, such a strategy does not always maximize the network lifetime [26]. Consider the target-tracking example again. While forwarding the gathered and processed data to the gateway, it is desirable to avoid routes through regions of the network that are running low on energy resources, thus preserving them for future, possibly critical detection and communication tasks. For the same reason, it is in general, undesirable to continuously forward traffic via the same path, even though it minimizes the energy, up to the point where the nodes on that path are depleted of energy, and the network connectivity is compromised. It would, instead, be preferable to spread the load more uniformly over the network. This general guideline can increase the network lifetime in typical scenarios, although this is not always the case [26] as the optimal distribution of traffic load is possible only when future network activity is known.

TOPOLOGY MANAGEMENT

The traffic distribution through appropriate routing essentially exploits the macro-scale redundancy of possible routes between source and destination. However, on each route, there is also a micro-scale redundancy of nodes that are essentially equivalent for the multi-hop path. In typical deployment scenarios, a dense network is required to ensure adequate coverage of both the sensing and multi-hop routing functionality, in addition to improving network fault-tolerance [11], [27]. It is immediately apparent that there exist several adaptive energy-fidelity tradeoffs here too. For example, in target tracking, denser distributions of sensors lead to increasingly precise tracking results. However, if network lifetime is more critical than tracking precision, tracking could be done using data samples from fewer nodes. In addition to reducing the computational complexity itself, this also reduces the communication requirements of the non- participating nodes since they no longer have to send in their data to be processed.

Despite the inherent node redundancy, these high densities do not immediately result in an increased network lifetime, as the radio energy consumption in *Idle* mode does not differ much from that in *Transmit* or *Receive* mode. Only by transitioning the radio to the *Sleep* state can temporarily quiescent nodes conserve battery energy. However, in this state, nodes cannot be communicated with, and have effectively retracted from the network, thereby changing the active topology. Thus, the crucial issue is to intelligently manage the sleep state transitions while providing robust undisturbed operation.

This reasoning is the foundation for the time slotted MAC protocol for sensor networks in [22] where the nodes only need to wake up during time slots that they are assigned to, although this comes at the cost of maintaining time synchronization. An alternative approach advocates explicit node wake up via a separate, but low-power paging channel. In addition, true topology management explicitly leverages the fact that in high node density several nodes can be considered backups of each other with respect to traffic forwarding. The GAF protocol [11] identifies equivalent nodes based on their geographic location in a virtual grid such that they replace each other directly and transparently in the routing topology. In SPAN [27], a limited number of coordinator nodes are elected to forward the bulk of the traffic as a backbone within the ad-hoc network, while other nodes can frequently transition to a sleep state. Both GAF and SPAN are distributed protocols that provision for periodic rotation of node functionality to ensure fair energy consumption distribution. STEM [28] goes beyond GAF and SPAN in improving the network lifespan by exploiting the fact that most of the time the network is only sensing its environment waiting for an event to happen. By eliminating GAF and SPAN's restriction of network capacity preservation at all times, STEM trades off an increased latency to set up a multi-hop path to achieve much higher energy savings.

COMPUTATION COMMUNICATION TRADEOFFS

Intelligent routing protocols and topology management ensure that the burden of forwarding traffic is distributed between nodes in an energy-efficient, *i.e.*, network lifetime improving, fashion. Further enhancements are possible by reducing the size of the packets that are forwarded. As mentioned earlier, each node already processes its sensor data internally to this end. Consider the target tracking application. Due to high node densities, a target is detected not only by a single node, but also by an entire cloud of nearby nodes, leading to a high degree of redundancy in the gathered data. Combining the information from the nodes in this cloud via in-network processing can both improve the reliability of the detection event/data, and greatly reduce the amount of traffic. One option is to combine the sensor readings of different nodes in a coherent fashion via beam-forming techniques [22]. Alternatively, non-coherent combining, also known as data fusion or aggregation, can be used, which does not require synchronization, but is less powerful. Several alternatives have been proposed to select the nodes that perform the actual combining, such as winner election [22], clustering [23], or traffic-steered [26]. These techniques illustrate the effectiveness of exploiting network wide computation-communication tradeoffs.

OVERHEAD REDUCTION

The sensor data packet payload can be quite compact due to in-network processing, with reported packet payloads as low as 8 to 16 bits [22]. Also, attribute based naming and routing are being used [3], where the more common attributes can be coded in fewer bits. Short random identifiers have been proposed to replace unique identifiers for end-to-end functions such as fragmentation/reassembly. Spatial reuse, combined with Huffman-coded representation, can significantly reduce the size of MAC addresses compared to traditional network-wide unique identifiers [21]. Packet headers using attribute-based routing identifiers and encoded reusable MAC addresses are very compact, of the order of 10 bits. This reduction will become more important as radios with smaller startup cost are developed [9].

CONCLUSIONS

Sensor networks have emerged as a revolutionary technology for querying the physical world and hold promise in a wide variety of applications. However, the extremely energy constrained nature of these networks necessitate that their design and operation be done in an energy-aware manner, enabling the system to dynamically make tradeoffs between performance, fidelity, and energy consumption. We presented several energy optimization and management techniques at node, link, and network level, leveraging which can lead to significant enhancement in sensor network lifetime.

ACKNOWLEDGEMENTS

REFERENCES

[1] Wireless Integrated Network Sensors, University of California, Los Angeles. (http://www.janet.ucla.edu/WINS)

[2] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: mobile networking for smart dust", in Proc. Mobicom, pp. 483-492, 1999.

[3] D. Estrin and R. Govindan, "Next century challenges: scalable coordination in sensor networks", in *Proc. Mobicom*, pp. 263-270, 1999.

[4] A. P. Chandrakasan, et al., "Design considerations for distributed microsensor systems", in *Proc. CICC*, 1999, pp. 279-286.

[5] J. Rabaey, et al., "PicoRadio supports ad hoc ultra low power wireless networking", in *IEEE Computer*, July 2000, pp. 42-48.

[6] A. P. Chandrakasan and R. W. Brodersen, Low Power CMOS Digital Design, Kluwer Academic Publishers, Norwell, MA, 1996.

[7] V. Tiwari, S. Malik, A. Wolfe, and M. T. C. Lee, "Instruction level power analysis and optimization of software", in *Jrnl. VLSI Signal Processing*, Kluwer Academic Publishers, pp. 1-18, 1996.

[8] A. Sinha and A. P. Chandrakasan, "Jouletrack: A web based tool for software energy profiling", in *Proc. Design Automation Conf.*, 2001.

[9] A. Wang, S-H. Cho, C. G. Sodini, and A. P. Chandrakasan, "Energy-efficient modulation and MAC for asymmetric microsensor systems", in *Proc. ISLPED*, 2001.

[10] WINS project, Rockwell Science Center, (http://wins..rsc.rockwell.com).

[11] Y. Xu, J. Heidemann and D. Estrin, "Geography-informed energy conservation for ad hoc routing", in *Proc. Mobicom*, 2001.

[12] C. F. Chiasserini and R. R. Rao, "Pulsed battery discharge in communication devices", in *Proc. Mobicom*, 1999.

[13] S. Park, A. Savvides, and M. Srivastava, "Battery capacity measurement and analysis using lithium coin cell battery", in *Proc. ISLPED*, 2001.

[14] L. Benini and G. DeMicheli, Dynamic Power Management: Design Techniques & CAD Tools, Kluwer Academic Publishers, Norwell, MA, 1997.

[15] T. A. Pering, T. D. Burd, and R. W. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms", in *Proc. ISLPED*, pp. 76-81, 1998.

[16] V. Raghunathan, P. Spanos, and M. Srivastava, "Adaptive power-fidelity in energy aware wireless embedded systems", to be presented at *IEEE Real Time Systems Symposium*, 2001.

[17] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy", in *Proc. Annual Symp. on Foundations of Computer Science*, pp.374-382, 1995.

[18] A. Sinha, A. Wang, and A. P. Chandrakasan, "Algorithmic transforms for efficient energy scalable computation", in *Proc. ISLPED*, 2000.

[19] V. Tsiatsis, S. Zimbeck, and M. Srivastava, "Architectural strategies for energy efficient packet forwarding in wireless sensor networks", in *Proc. ISLPED*, 2001.

[20] C. Schurgers, O. Aberthorne, and M. Srivastava, "Modulation scaling for energy aware communication systems", in *Proc. ISLPED*, 2001.

[21] C. Schurgers, G. Kulkarni, and M. Srivastava, "Distributed assignment of encoded MAC addresses in wireless sensor networks", in *Proc.MobiHoc*, 2001.

[22] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network", in *IEEE Personal Comm. Magazine*, vol.7, no.5, pp. 16-27, Oct. 2000.

[23] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks", in *Proc. Hawaii Intl. Conf. on System Sciences*, Hawaii, 2000.

[24] P. Lettieri, C. Fragouli, and M. Srivastava, "Low power error control for wireless links", in *Proc. Mobicom*, pp. 139-150, 1997.

[25] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks", in *Proc. INFOCOM*, 2000.

[26] C. Schurgers and M. Srivastava, "Energy efficient routing in sensor networks", in *Proc. Milcom*, 2001.

[27] B. Chen, K. Jamieson, H. Balakrishnan, and R Morris, "SPAN: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, in *Proc. Mobicom*, 2001.

[28] C. Schurgers, V. Tsiatsis, and M. Srivastava, "STEM: Topology management for energy efficient sensor networks," To appear in the Proceedings of the 2002 IEEE Aerospace Conference, March 2002.
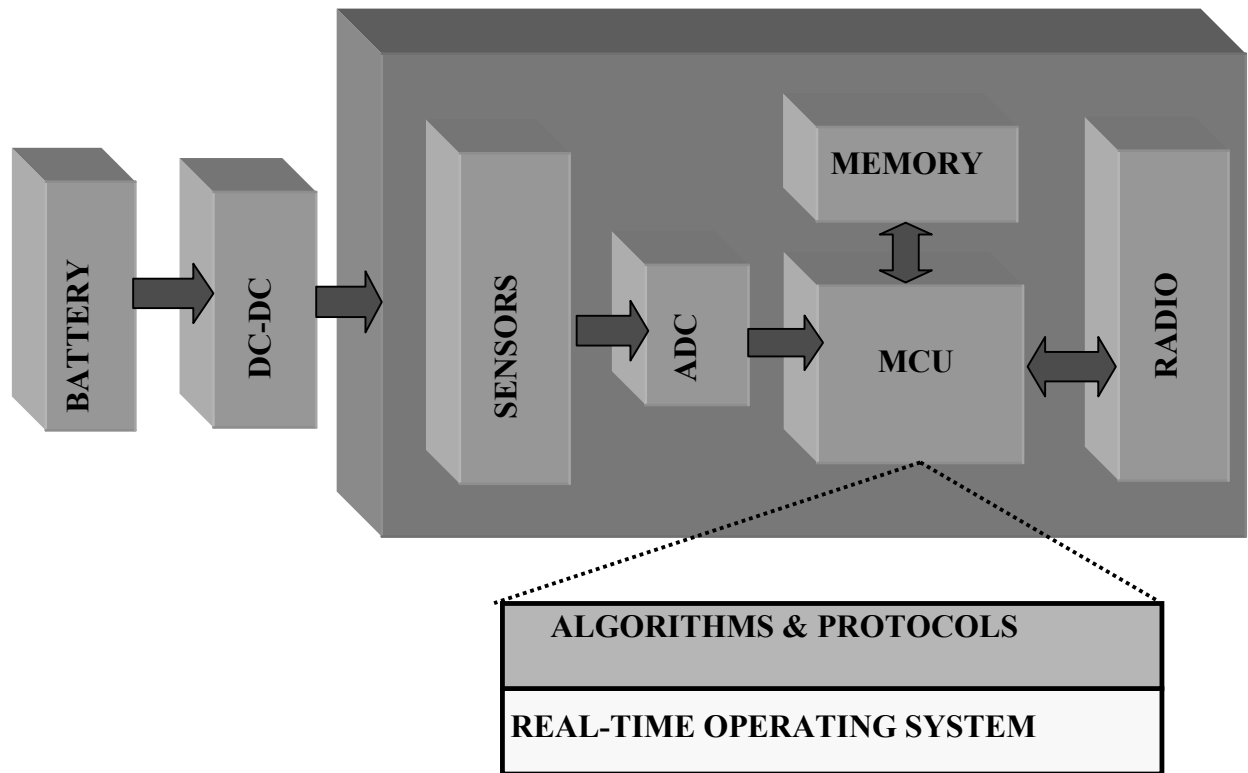
Fig 1. System architecture of a typical wireless sensor node
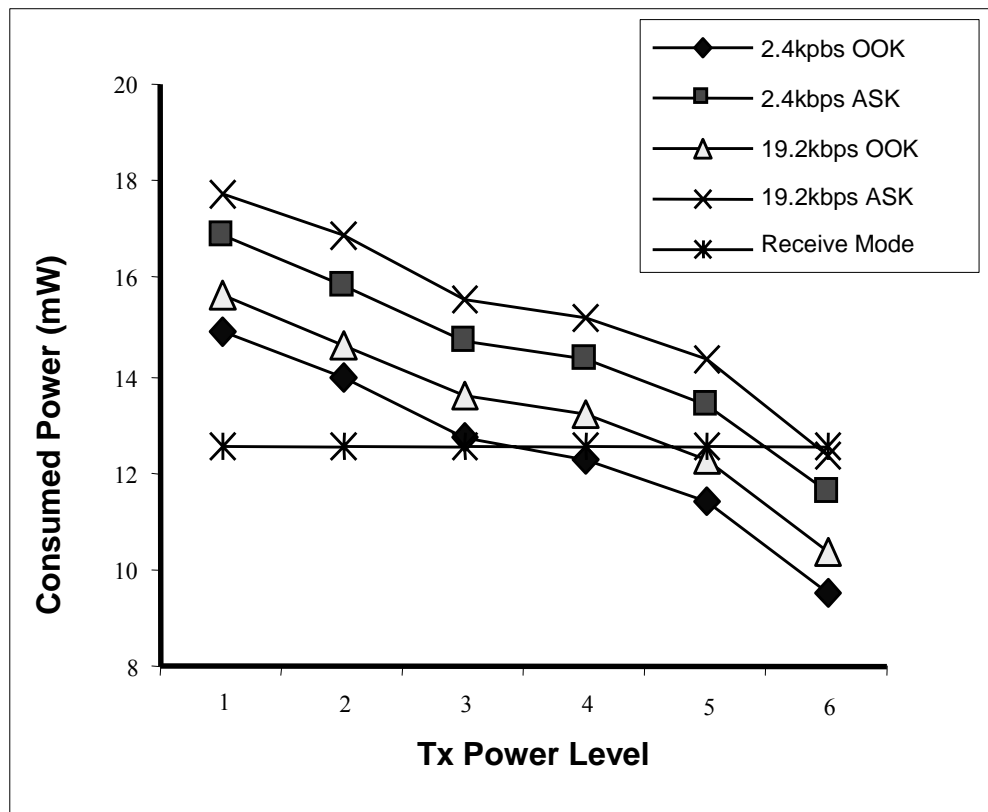
Fig. 2. Power consumption of an RFM radio in various modes of operation
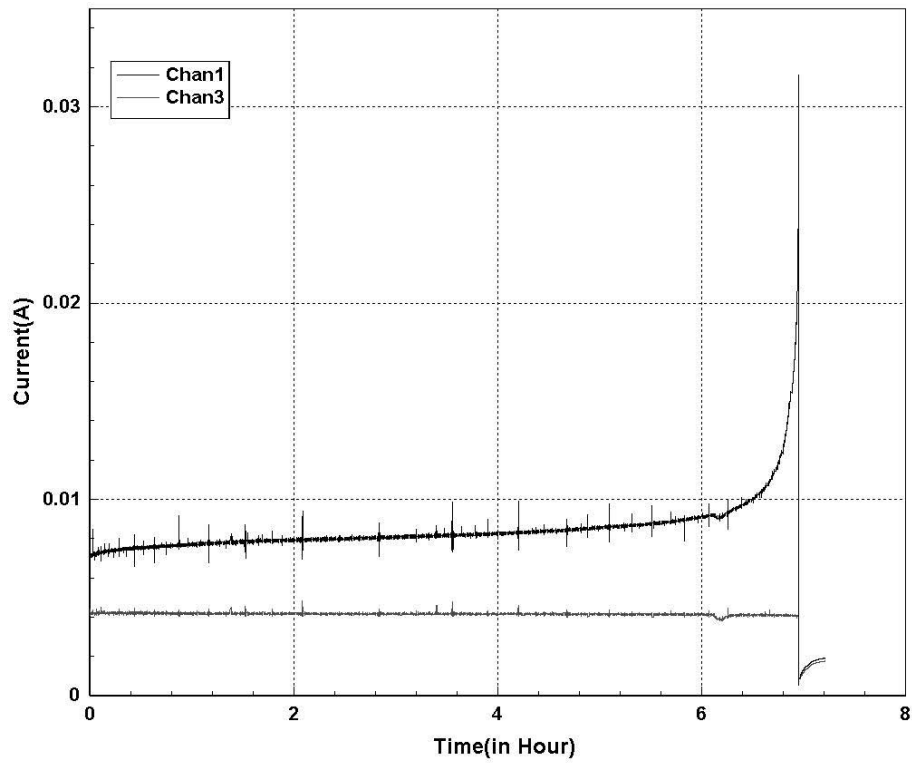
Fig. 3. Current drawn from the battery (Chan 1) and current supplied to the node (Chan 3)
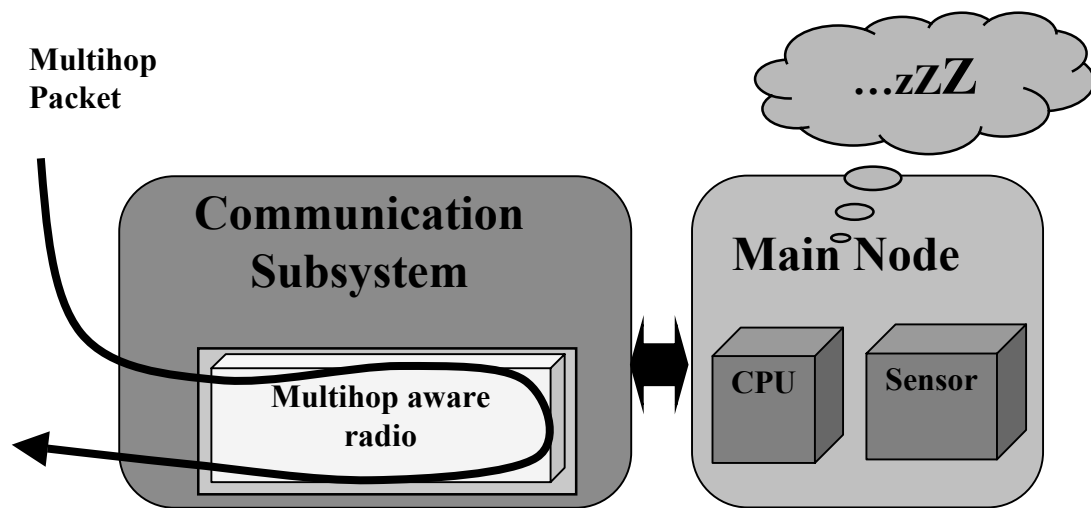
Fig. 4. Energy-aware packet forwarding architecture
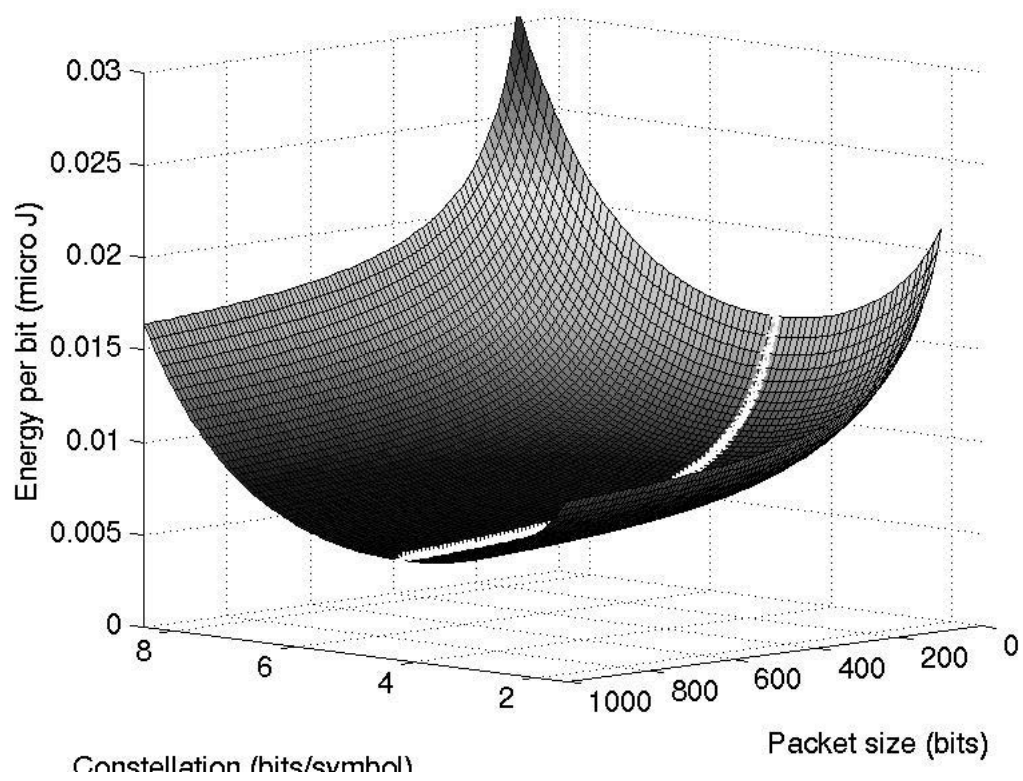
Fig. 5. Radio energy per bit as a function of packet size and modulation level
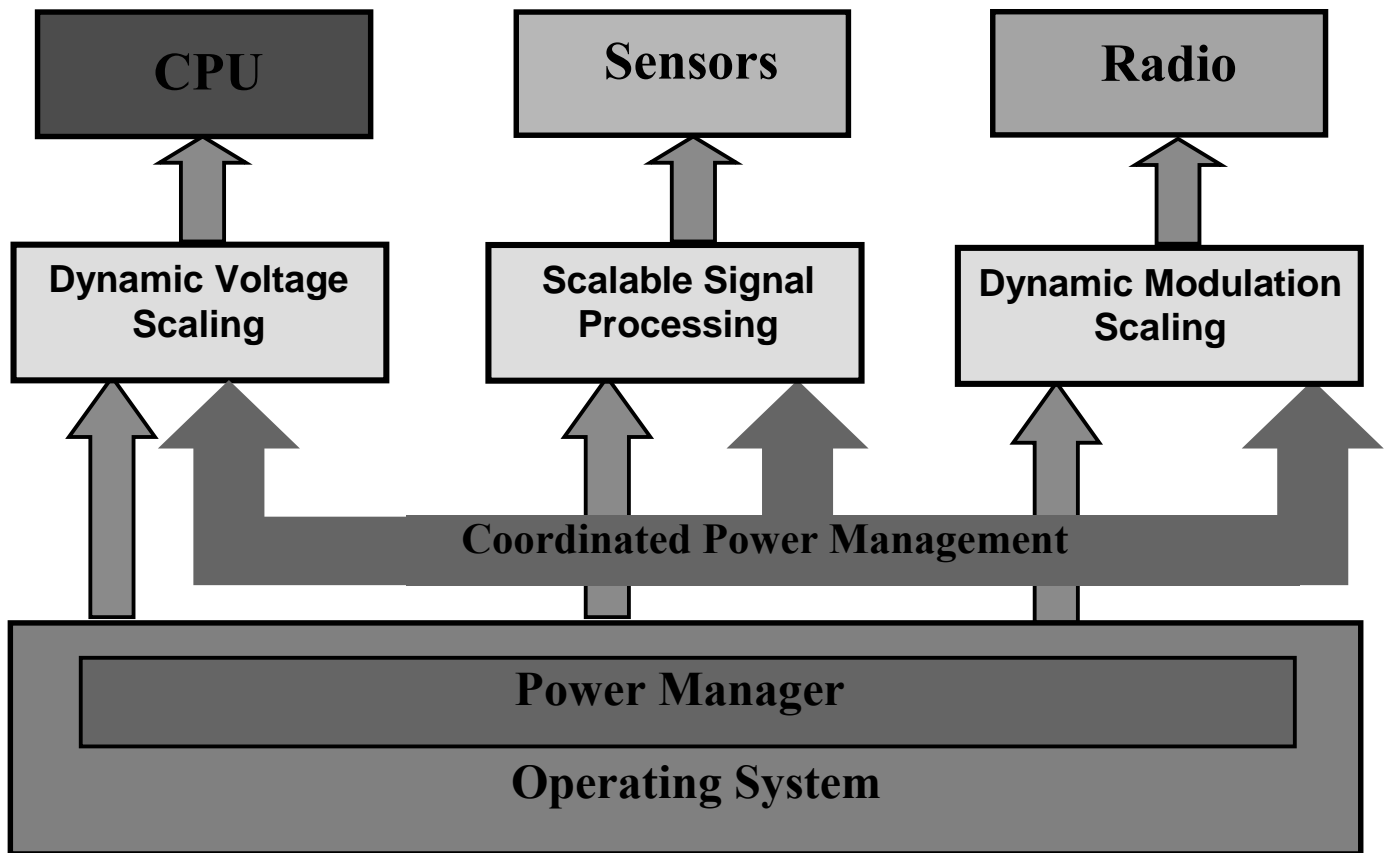
Fig. 6. Coordinated power management at the node level to exploit computation-communication tradeoffs

TABLE I

POWER ANALYSIS OF ROCKWELL'S WINS NODES

MCU Mode	Sensor Mode	Radio Mode	Power (mW)
Active	On	Tx (Power: 36.3 mW)	1080.5
		Tx (Power: 19.1 mW)	986.0
		Tx (Power: 13.8 mW)	942.6
		Tx (Power: 3.47 mW)	815.5
		Tx (Power: 2.51 mW)	807.5
		Tx (Power: 0.96 mW)	787.5
		Tx (Power: 0.30 mW)	773.9
		Tx (Power: 0.12 mW)	771.1
Active	On	Rx	751.6
Active	On	Idle	727.5
Active	On	Sleep	416.3
Active	On	Removed	383.3
Sleep	On	Removed	64.0
Active	Removed	Removed	360.0

TABLE II

POWER ANALYSIS OF MEDUSA II NODES

MCU Mode	Sensor Mode	Radio Mode	Mod. Scheme	Data Rate	Power (mW)
Active	On	Tx (Power: 0.7368 mW)	OOK	2.4 kbps	24.58
		Tx (Power: 0.0979 mW)	OOK	2.4 kbps	19.24
		Tx (Power: 0.7368 mW)	OOK	19.2 kbps	25.37
		Tx (Power: 0.0979 mW)	OOK	19.2 kbps	20.05
		Tx (Power: 0.7368 mW)	ASK	2.4 kbps	26.55
		Tx (Power: 0.0979 mW)	ASK	2.4 kbps	21.26
		Tx (Power: 0.7368 mW)	ASK	19.2 kbps	27.46
		Tx (Power: 0.0979 mW)	ASK	19.2 kbps	22.06
Active	On	Rx	Any	Any	22.20
Active	On	Idle	Any	Any	22.06
Active	On	Off	Any	Any	9.72
Idle	On	Off	Any	Any	5.92
Sleep	Off	Off	Any	Any	0.02

TABLE III

TYPICAL RADIO PARAMETERS FOR SENSOR NETWORKS

E_{start}	1∞J
P_{elec}	12mW
P_{RF}	1mW for 4-QAM
R_S	1 Mbaud
H	16 bits

E^2WFQ: An Energy Efficient Fair Scheduling Policy For Wireless Systems

Vijay Raghunathan, Saurabh Ganeriwal, Curt Schurgers, and Mani Srivastava

Networked and Embedded Systems Lab (NESL)
Department of Electrical Engineering
University of California, Los Angeles

{vijay, saurabh, curts, mbs}@ee.ucla.edu

ABSTRACT

As embedded systems are being networked, often wirelessly, an increasingly larger share of their total energy budget is due to the communication. This necessitates the development of power management techniques that address communication subsystems, such as radios, as opposed to computation subsystems, such as embedded processors, to which most of the research effort thus far has been devoted. In this paper, we present E^2WFQ, an energy efficient version of the Weighted Fair Queuing (WFQ) algorithm for packet scheduling in communication systems. We employ a recently proposed radio power management technique, Dynamic Modulation Scaling (DMS), as a control knob to enable energy-latency tradeoffs during wireless packet scheduling. The use of E^2WFQ results in an energy aware packet scheduler, which exploits the statistics of the input arrival pattern as well as the variability in packet lengths. Simulation results show that large savings in energy consumption can be obtained through the use of our scheduling scheme, compared to conventional WFQ, with only a small, bounded increase in worst case packet latency.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications, Wireless communication*; C.2.6 [**Computer-Communication Networks**]: Internetworking—*Routers*

General Terms

Algorithms, Design

Keywords

Energy Efficient Design, Power Management, Wireless Communications, Fair Scheduling

1. INTRODUCTION

Conventional low power design techniques [1, 2] and hardware architectures [3] only target digital computation systems, and rela-
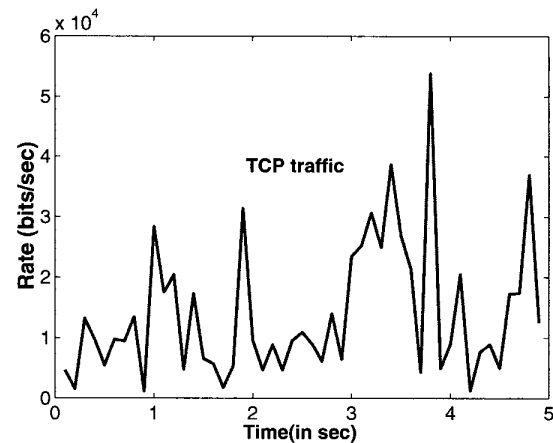
Figure 1: A 5 second workload trace at a network router

tively little work has been done for power optimization of the wireless communication subsystem. In many wireless embedded systems, communication energy dominates the energy consumed for computation [4], accentuating the need for radio power management methodologies. For example, in the wireless sensor nodes from Rockwell Inc. [5], transmitting one bit consumes 1500 to 2700 times [4] (depending on the transmission range) as much energy as executing one instruction. Therefore, in wireless devices, power management cannot just be limited to computation subsystems, such as processors, but has to be extended to communication subsystems, such as radios, as well.

Radio power management is, however, complex, since the dependence of radio energy consumption on supply voltage and other circuit parameters is weak. Existing power management techniques that are effective for computation subsystems, such as Dynamic Voltage Scaling (DVS) [6], therefore do not result in the required energy savings. However, there exist similar control knobs on the radio that can be exploited for power management. The recently proposed technique of Dynamic Modulation Scaling (DMS) [7] uses the modulation level as an energy-speed control knob that can be fine-tuned to enable dynamic power-performance tradeoffs in the communication system (similar to what DVS provides for digital circuits [6]).

As is the case with variable voltage computation systems, significant energy benefits can be achieved in wireless communication systems by recognizing that peak performance (*i.e.*, service

rate) is not always required. Since network traffic is characterized by a time varying workload requirement, energy can be saved by dynamically adapting the output transmission rate accordingly, through the use of DMS. Figure 1 shows a 5 second snapshot of a real workload trace for a TCP traffic stream at a network router [8], exemplifying the inherent workload variability. For effective system-level power management, we need to develop algorithms that can exploit this variability by using control knobs such as DMS.

1.1 Paper contributions

In this paper, we present an energy aware version of the Weighted Fair Queuing (WFQ) [9] scheduling policy for communication systems, which we call E^2WFQ. Our algorithm results in an energy aware packet scheduler, which is capable of operating at hitherto unreachable points on the energy-latency tradeoff curve. Our previous work [10] showed the energy benefits of using DMS in the context of a deadline based real-time scheduling scheme. However, WFQ based packet schedulers are far more widely used than deadline based packet schedulers (both in wired and wireless environments) due to their desirable properties, which are elaborated further in Section 2.1. Therefore, the techniques presented in this paper find applicability in enhancing the energy awareness of a much larger class of communication systems. To the best of our knowledge, this is the first work that attempts to incorporate energy awareness into rate based fair scheduling.

1.2 Related work

Several power management techniques have been proposed for the energy efficient design and operation of computation subsystems. One of the most effective techniques is DVS [6], where workload variability is exploited for energy savings by dynamically adjusting the processor's supply voltage and clock frequency to match the instantaneous performance requirement. Numerous energy aware task scheduling schemes have also been proposed, which utilize DVS to yield significant energy savings. Variable voltage task scheduling for base station like environments has been explored in [11, 12], while the work described in [13, 14, 15, 16, 17, 18] deals with energy aware real-time task scheduling.

Unlike the large body of work that exists on variable voltage task scheduling, relatively little work has been done for energy efficient wireless packet scheduling. Fair packet scheduling has been an active research topic in the networking community for a long time. Several fair scheduling schemes have been proposed for both wired (e.g., Weighted Round Robin [19], Start-Time Fair Queuing [20], Worst-Case Fair Weighted Fair Queuing [21]) and wireless (e.g., Channel-State Independent Wireless Fair Queueing [22], Wireless Packet Service [23], Server Based Fairness Approach [24]) environments. However, all of them are based on the concept of Generalized Processor Sharing [25], and its packetized version, Packet-by-Packet Generalized Processor Sharing [26] (PGPS, also known as WFQ [9]). Since network traffic usually displays a high temporal variation, the leaky bucket mechanism [25] is used as a standard way to regulate input streams, and control their extent of variability. Finally, low power medium access protocols based on packet scheduling for wireless ATM networks were proposed in [27].

2. BACKGROUND

2.1 Generalized Processor Sharing (GPS)

Since GPS is different from conventional CPU task scheduling disciplines, we first review some of its basic concepts. A GPS scheduler divides the total link capacity C, among N input streams according to their service requirements. Each stream i is character-

ized by a weight ϕ_i, such that its service rate, g_i, is guaranteed to be [25][1]:

$$g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} \times C \qquad (1)$$

GPS has the following attractive features: (i) Different input streams are well-isolated from each other (since each of them is allocated a guaranteed rate). Therefore, a malicious input stream cannot starve other streams of link bandwidth by generating large amounts of traffic. (ii) By varying the ϕ_i's, we have the flexibility of changing the fraction of the output link bandwidth that is allocated to each stream. As long as the combined average input rate of all the streams is less than C, any positive assignment of ϕ_i's yields a stable system.

2.2 Realizable implementations of GPS: WFQ

The GPS service model is an ideal one in which the traffic is considered to be infinitely divisible, and all streams are served simultaneously. Although GPS cannot be realized in practice, several real implementations of packet schedulers (for wired, as well as wireless environments) are based on it, and try to emulate the service provided by GPS as closely as possible.

Let F_p be the time at which a packet would complete service under GPS. Then, a good approximation of GPS is a scheme that serves packets with earliest F_p first. This scheme was proposed independently by two groups of researchers as Weighted Fair Queuing [9], and Packetized GPS [26], respectively. The difference in packet delays between GPS and WFQ was shown to be bounded by $\frac{L_{max}}{C}$, where L_{max} is the maximum packet size, and C is the output rate of the system [26].

2.3 Leaky bucket mechanism

The leaky bucket mechanism is often used to regulate and police the traffic in a network. This model is attractive since it restricts the incoming traffic in terms of average rate, as well as burstiness. For each stream i, the amount of traffic, $A_i(\tau, t)$ that enters the network in time interval $(\tau, t]$ conforms to [25]:

$$A_i(\tau, t) \leq \sigma_i + \lambda_i \times (t - \tau), \forall t \geq \tau \geq 0 \qquad (2)$$

where λ_i is a parameter characterizing the average rate of stream i, and σ_i is a parameter characterizing its burstiness. When the input sources are constrained by leaky buckets, it is possible to give a worst case queuing delay guarantee using GPS (and therefore, WFQ).

Theorem 1: *If the input traffic of stream i is constrained using a leaky bucket with parameters (σ_i, λ_i), and g_i is its guaranteed rate, then the maximum delay for a packet of stream i, under GPS scheduling is given by [26]:*

$$D_i \leq \frac{\sigma_i}{g_i} \qquad (3)$$

2.4 Dynamic Modulation Scaling

We next review the basics of DMS, which was introduced in [7]. To transmit information, bits are coded into channel symbols. The
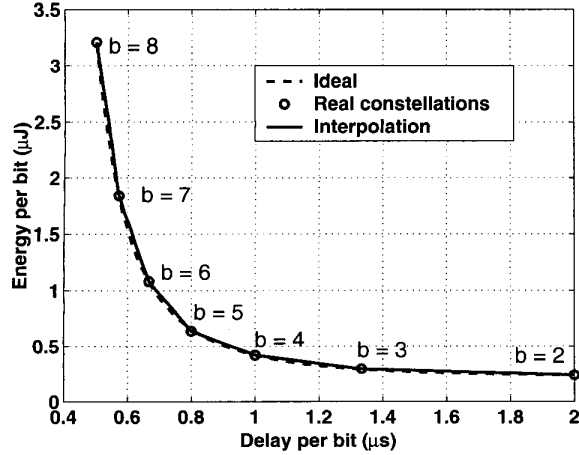
146

Figure 2: Energy-Delay tradeoff obtained through DMS

number of bits per symbol is given by the modulation level b. This b is the radio control knob that allows DMS to trade off energy versus delay. The average time to transmit one bit is given by Equation (4), where R_S is the symbol rate in number of symbols sent over the channel per second.

$$T_{bit} = \frac{1}{b \times R_S} \qquad (4)$$

Although DMS is applicable to other scalable modulation schemes as well, we focus on Quadrature Amplitude Modulation (QAM) as it is both efficient and easy to implement [28]. The energy consumed for transmitting one bit is given by [7]:

$$E_{bit} = C_S \times \frac{2^b - 1}{b} + C_E \times \frac{1}{b} \qquad (5)$$

The first term gives the energy consumed in the radio front-end for generating the electro-magnetic waves that carry the information. Parameter C_S depends on the radio implementation, the wireless channel, the transmit distance, and the required error performance. Strictly speaking, it is also a function of b, but a very weak one [7], and therefore, we assume it to be a constant. The rest of the radio energy consumption is lumped into the second term of Equation (5), where C_E depends on the radio implementation. Although Equation (5) is only exact for *even* integer values of b, it is also a reasonable approximation when b is *odd*. In addition, a packet can be split into two parts, each with a different modulation. In this case, the average energy and delay per bit for the packet as a whole are a linear interpolation between the corresponding values of the two modulation levels.

Figure 2 plots the energy versus delay for the following parameter values: R_S = 250 KHz, C_S = 100 nJ, and C_E = 180 nJ. The values of C_S and C_E are extracted from [29], which describes the implementation of an adaptive QAM system[2]. Figure 2 shows the operating points that correspond to real constellations and those that are obtained through interpolation. The curve labeled *Ideal* is calculated from the above equations. Each modulation scheme

---

[2]Since the system in [29] was designed for high speed rather than low power applications, it is likely that these numbers can be reduced further through the use of dedicated circuit design techniques.
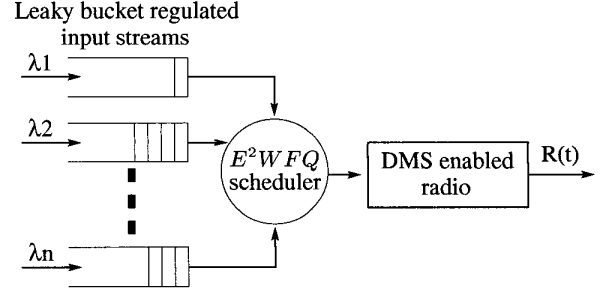
Figure 3: E^2WFQ **scheduler serving leaky bucket regulated streams**

has a b_{min}, which is equal to 2 for QAM. The maximum modulation b_{max} is only bounded by implementation constraints. In our work, we choose b_{max} equal to 8, and scale the modulation level with a granularity of 0.5 bits/symbol. When the sender changes the modulation, the receiver needs to be told. To avoid a complicated sender-receiver protocol for modulation changes, and to limit the associated overhead, we restrict these changes to be done only at the start of packet transmissions. This finite time-granularity is a crucial difference between DMS and dynamic voltage scaling [6].

3. THE E^2WFQ ALGORITHM

Having explained the basics of rate based fair scheduling and DMS, we next present our algorithm, E^2WFQ, which uses DMS to incorporate energy awareness into rate based scheduling. Figure 3 shows a generic block diagram of our system, which consists of an E^2WFQ scheduler followed by a radio that can perform DMS. The scheduling technique can be used either for multiple streams sharing a point-to-point link, or for multiple streams destined to different one-hop neighbors of a wireless node (*e.g.*, a base station sending data to multiple clients). Each stream i is regulated by a leaky bucket with parameters (σ_i, λ_i). Therefore, each of the N input streams produces packets at an average rate of λ_i. The average rate at which packets arrive at the scheduler is $\lambda = \sum_{i=1}^{N} \lambda_i$. The adaptive modulation radio ensures that the scheduler operates at a time varying output rate of $R(t)$. Stream i is allocated a weight ϕ_i, which leads to a corresponding guaranteed rate, g_i.

3.1 Energy saving opportunities

In many practical scenarios, the average input rate of a stream, λ_i, is lower than its guaranteed rate, g_i, resulting in a low link utilization. In addition, packet lengths may often be smaller than the maximum value, thereby reducing the utilization even further. Therefore, it is possible to operate at an instantaneous output rate, $R(t)$ that is lower than the maximum rate, C, for most of the time. So, instead of operating at C, and shutting down the radio when idle, we slow transmissions down to a rate $R(t)$. As can be seen in Figure 2, the energy-delay curve is convex, which implies that slowing down the radio is more energy efficient than shutdown (similar to what is observed in DVS [6]).

3.2 Overview of the problem

The goal of E^2WFQ is to adapt the instantaneous output rate $R(t)$ to match the instantaneous workload. At the same time, we would like to bound the performance impact that may result. Due to the convexity of the energy-speed curve and Jensen's inequality $\left(\overline{E(r)} \leq E(\overline{r}) \right)$, workload averaging results in higher energy

147

savings. In fact, maximum energy savings would be obtained if we operated at the long term average input rate, thereby smoothing out all the input workload variations. However, buffering the input variations leads to a performance penalty due to an increase in packet delays. Therefore, the crux of the problem reduces to determining the degree of buffering (*i.e.*, how much workload averaging to perform) while still bounding the increase in packet delays.

3.3 Monitoring the input rate

We observe that the instantaneous queue size (*i.e.*, number of packets in the queue) is a good indicator of the instantaneous arrival rate. If the input rate suddenly becomes greater than the output rate, the queue size increases. On the other hand, if the input rate suddenly drops below the output rate, the queue gets drained. We introduce a new parameter Δ, which determines the system's response to workload variations, by deciding the degree of buffering. As we will see shortly, Δ also determines the maximum impact that our scheme can have on packet delay.

Definition 1: *We define Δ to be the desired time from a packet's arrival at the end of the queue to its departure from the head of the queue.*

We can see that if the input rate remains constant, Δ is the delay experienced by every packet in the queue.

3.4 Computing the required output rate

Whenever a packet arrives, the required rate of the particular stream is recomputed. Assume that the newly arrived packet is the m'th packet in the queue, and the current time is τ. Let the arrival times of the m packets in the queue be given by $A_1, A_2, \ldots, A_m$ (where $A_m = \tau$). Then, the required rate $(r_{i,k})$ for the k'th packet to have a total delay of Δ is given by:

$$r_{i,k} = \frac{k \times L_i}{(A_k + \Delta - \tau)} \quad \forall k \in \{1, \ldots, m\} \quad (6)$$

where L_i is the length of a packet belonging to stream i. Two observations are apparent from Equation (6), (i) The required rate for the newly arrived packet (*i.e.*, last packet in the queue) is $\frac{m \times L_i}{\Delta}$, and (ii) If the output rate is equal to the required rate, then the required rate of this packet remains constant as it progresses through the queue.

The required rate for a stream $R_{out,i}$ is the minimum rate at which Equation (6) is satisfied for all the m packets in the queue. However, to ensure isolation of input streams, this required rate cannot be greater than g_i. Therefore, we have:

$$R_{out,i} = Min \ \{Max \ (r_{i,1}, r_{i,2}, \ldots, r_{i,m}), \ g_i\} \quad (7)$$

Summing the required rate over all the streams, we get the instantaneous required rate of the system to be $R = \sum_{i=1}^{N} R_{out,i}$.

3.5 Setting the output link speed

Although the maximum output link rate is C, the instantaneous required rate by all the streams together is only $R = \sum_{i=1}^{N} R_{out,i}$. This means that the output link can be slowed down to just meet the instantaneous requirement, thus saving energy. The new modulation level for the outgoing packets is given by:

$$b_{instantaneous} = \frac{R}{C} \times b_{max} \quad (8)$$

3.6 Accounting for variable packet sizes

Quite often, packet lengths in network traffic can be variable. This is especially true for packets generated by multimedia streams
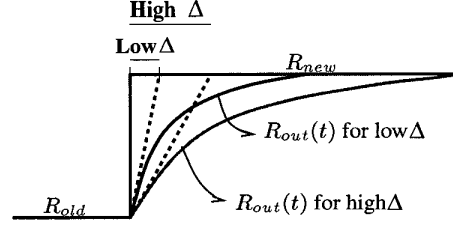
Figure 4: The effect of Δ on the output rate

such as variable bit rate audio and video codecs. To exploit the variation in packet lengths to further scale down the modulation level and obtain more energy savings, we generalize the calculation of a packet's required rate. If the length of the k'th packet in the queue is $L_{i,k}$, and there are a total of m packets in the queue, then the following equation is used to compute the required rate of a packet, instead of Equation (6):

$$r_{i,k} = \frac{\sum_{j=1}^{k} L_{i,j}}{(A_k + \Delta - \tau)} \quad \forall k \in \{1, \ldots, m\} \quad (9)$$

3.7 Delay guarantee provided by E^2WFQ

Through the choice of the parameter Δ, our scheduling scheme provides the following delay bound for packets.

Theorem 2: *The maximum delay of a packet of stream i, under the E^2WFQ scheduling scheme is given by:*

$$D_i^* \leq \left(D_i + \Delta + \frac{L_{max}}{C_{min}} \right) \leq \left(\frac{\sigma_i}{g_i} + \Delta + \frac{L_{max}}{C_{min}} \right) \quad (10)$$

where C_{min} is the output link capacity at a modulation level b_{min}. **Proof:** The proof follows directly from Theorem 1, the bounded difference in packet delays between GPS and WFQ, and the definition of Δ.

3.8 Analyzing the system's response

We next analyze the system's response to a change in workload to highlight the effect of Δ. Let the input rate for stream i be equal to R_{old}. If we set the output rate to be equal to R_{old}, then the queue soon reaches a steady state, and the number of packets in the queue remains constant. Let the system be at such a steady state at time t, and the number of packets in the queue be $y(t)$. In this state, any packet that arrives, sees exactly $y(t) - 1$ packets ahead of it in the queue, each of length, say L_i. All these packets (including the one that just arrived) have to complete transmission within a time Δ. Therefore, the output rate, $R_{out}(t)$ is given by (see first paragraph after Equation (6)):

$$R_{out}(t) = \frac{y(t) \times L_i}{\Delta} = R_{old} \quad (11)$$

Now, let the input rate increase abruptly to R_{new}. We want to analyze how fast the system stabilizes to its new steady state. After a time δt, the number of packets in the queue becomes:

$$y(t + \delta t) = y(t) + \frac{R_{new} - R_{out}(t)}{L_i} \times \delta t \quad (12)$$

The new output rate will be given by $R_{out}(t + \delta t)$, which after

148
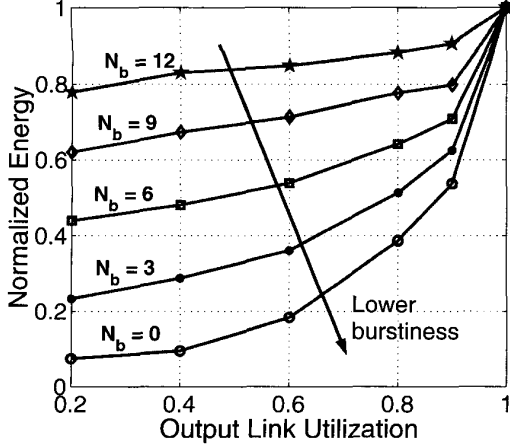
Figure 5: Normalized energy consumption of E^2WFQ, as a function of link utilization, for varying degrees of burstiness

Figure 6: One second snapshot showing how the output rate follows the input rate

some simple substitutions, can be written as:

$$R_{out}(t + \delta t) = R_{out}(t) + \frac{R_{new} - R_{out}(t)}{\Delta} \times \delta t \qquad (13)$$

Therefore, the rate at which the output rate changes is given by:

$$\frac{\delta R_{out}}{\delta t} = \frac{1}{\Delta} \times (R_{new} - R_{out}(t)) \qquad (14)$$

Solving this differential equation, and applying the initial condition $R_{out}(t) = R_{old}$, we get:

$$R_{out}(t) = R_{new} + (R_{old} - R_{new}) \times e^{\frac{-t}{\Delta}} \qquad (15)$$

The evolution of $R_{out}(t)$ is shown in Figure 4. Therefore, it can be concluded that Δ is the time constant of the first-order input response of the system. A high value of Δ means that the system takes longer to switch to the new steady state, which implies that steep workload transients are filtered out. While this increases the energy savings, the maximum impact on packet delay also increases (see Equation (10)). On the other hand, a low value of Δ means that the system is sensitive to changes in the input rate, and performs lesser workload averaging. This provides lower energy savings, but also results in a smaller impact on packet delay. The best choice of Δ thus depends on the allowable delay, and the input rate variability. A detailed investigation of the optimal choice of Δ is beyond the scope of this paper.

4. SIMULATION RESULTS

To evaluate the impact of our techniques, we carried out a number of simulations. In the following subsections, we describe our simulation framework, and present our results.

4.1 Simulation framework

We used a C based discrete event simulator, PARSEC [30], and considered a network scenario where 4 streams share an output wireless link. The values of R_S and b_{max} are the same as in Section 2.4, resulting in a total link capacity $C = 2$ Mbit/s. The streams are allocated equal weights such that each stream is guaranteed a rate of $\frac{C}{4} = 500$ Kbit/sec. The maximum packet size is set to be L_{max} = 1000 bits, and each stream is leaky bucket constrained with the same maximum burstiness parameter $\sigma_i = 100$ packets. For our
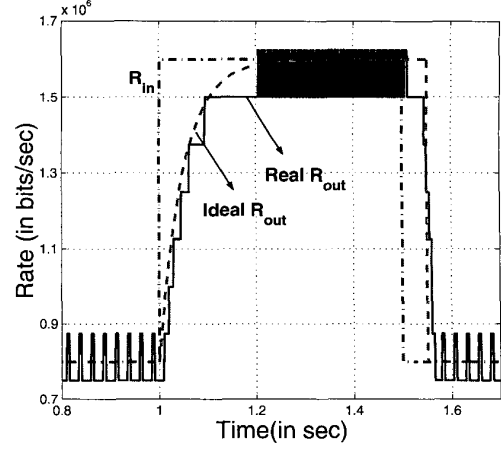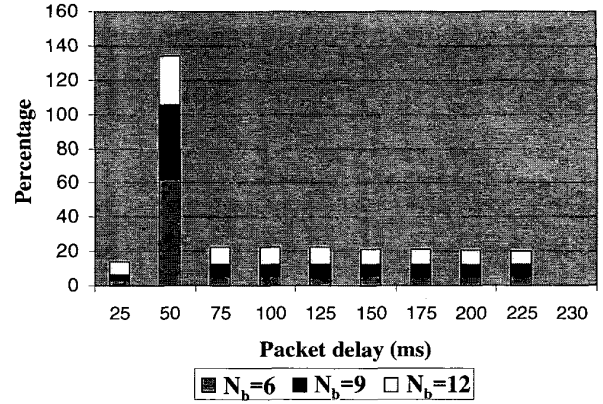
Figure 7: Packet delay histogram for three different levels of burstiness

values, D_i is equal to 0.2 seconds. The average rate of the input is varied to change the link utilization. As explained in the previous section, the choice of Δ offers a tradeoff between energy savings and queuing delay. In our work, we have set $\Delta = 0.05$ seconds.

4.2 Discussion of the results

Figure 5 shows the energy consumed by E^2WFQ, normalized against the energy consumed by conventional WFQ, for varying values of output link utilization. As expected, the energy consumption of E^2WFQ decreases as the link utilization drops. Our algorithm reduces to conventional WFQ at a utilization of one, since there are no opportunities for slowing down. The different curves in Figure 5 show the energy consumption for varying degrees of burstiness (variability) in the input workload. N_b gives the number of equally spaced bursts appearing at a stream's input over the observation window of 1000 packets of the stream. A high value of N_b indicates a high degree of burstiness. For a given link utilization, increasing the level of input burstiness increases the energy consumption, which is intuitive, since bursts at the input cause the output rate to increase, thus decreasing the energy savings. Figure 6 shows how the output rate follows the input rate, for our choice

149

of Δ = 0.05 seconds. The curve marked *Ideal R_{out}* shows the curve predicted by Equation (15). The actual output rate can only increase in increments of $\frac{L_i}{\Delta}$, which is clearly seen in the curve marked *Real R_{out}* in the figure.

Figure 7 shows the distribution of packet delays for varying levels of burstiness for a fixed link utilization. The bar at x = x_i indicates the percentage of packets whose delays lie in the interval $(x_{i-1}, x_i]$. There are three important observations that can be made from this figure: (i) Since the input rate fluctuates around its average rate, most of the packets experience a delay equal to Δ. (ii) As the level of burstiness increases, the percentage of packets that experience a delay larger than Δ also increases, and (iii) None of the packets violate the delay bound given in Theorem 2. Our results (Figures 5 - 7) show that E^2WFQ achieves a significant reduction in energy consumption, at the cost of a relatively small increase in packet latencies.

5. CONCLUSIONS

In wireless embedded systems, communication energy is increasingly becoming the dominant component of total energy consumption. Power management techniques therefore, should also address communication subsystems such as radios, as opposed to only computation subsystems such as embedded processors. DMS is a technique that offers a power-speed control knob for the radio, and can thus be used for communication power management. Higher level techniques are needed that can exploit this control knob to enable system level energy-latency tradeoffs. In this paper, we have targeted the packet scheduling process and investigated avenues for making it energy aware. We have presented E^2WFQ, an energy efficient version of the Weighted Fair Queuing (WFQ) algorithm for packet scheduling in communication systems. Our algorithm results in an energy aware packet scheduler that provides significant energy savings (up to a factor of 10) with only a small increase in worst case packet delay.

ACKNOWLEDGEMENTS

6. REFERENCES

[1] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, Norwell, MA, 1998.

[2] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, Norwell, MA, 1997.

[3] A. P. Chandrakasan and R. W. Brodersen, *Low Power CMOS Digital Design*. Kluwer Academic Publishers, Norwell, MA, 1996.

[4] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy aware wireless microsensor networks", *IEEE Signal Processing Magazine*, vol. 19, iss. 2, pp. 40-50, March 2002.

[5] Rockwell Inc. (http://wins.rsc.rockwell.com)

[6] T. A. Pering, T. D. Burd, and R. W. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms", *Proc. ACM ISLPED*, pp. 76-81, 1998.

[7] C. Schurgers, O. Aberthorne, and M. B. Srivastava, "Modulation scaling for energy aware communication systems", *Proc. ACM ISLPED*, pp. 96-99, 2001.

[8] The Internet Traffic Archive (http://ita.ee.lbl.gov).

[9] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorthm", *Internet Res. and Exper.*, vol. 1, 1990.

[10] C. Schurgers, V. Raghunathan, and M. B. Srivastava, "Modulation scaling for real-time energy-aware packet scheduling", *Proc. IEEE GLOBECOM*, pp. 3653-3657, 2001.

[11] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy", *Proc. First Symp. on OSDI*, pp. 13-23, 1994.

[12] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU", *Proc. ACM MOBICOM*, pp. 13-25, 1995.

[13] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors, *Proc. ACM ISLPED*, pp. 197-202, 1998.

[14] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processors", *Proc. IEEE ICCAD*, pp. 653-656, 1998.

[15] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems", *Proc. ACM DAC*, pp. 134-139, 1999.

[16] V. Raghunathan, P. Spanos, and M. B. Srivastava, "Adaptive power-fidelity in energy-aware wireless systems", *Proc. IEEE RTSS*, pp. 106-115, 2001.

[17] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low power embedded operating systems", *Proc. SOSP*, pp. 89-102, 2001.

[18] F. Gruian, "Hard real-time scheduling for low energy using stochastic data and DVS processors", *Proc. ACM ISLPED*, pp. 46-51, 2001.

[19] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin", *Proc. IEEE SIGCOMM*, pp. 231-242, 1995.

[20] P Goyal, H. M. Vin, and H. Cheng, "Start-time fair queuing: A scheduling algorithm for integrated servicespacket switching networks", *Proc. IEEE SIGCOMM*, pp. 157-168, 1996.

[21] J. C. R. Bennett and H. Zhang, "WF^2Q : Worst-case fair weighted fair queueing, *Proc. IEEE INFOCOM*, pp. 120-128, 1996.

[22] P. Lin, B. Bensaou, Q. L. Ding, and K. C. Chua, "A wireless fair scheduling algorithm for error-prone wireless channels", *Proc. ACM WoWMOM*, pp. 11-20, 2000.

[23] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks", *Proc. IEEE SIGCOMM*, pp. 63-74, 1997.

[24] P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks", *Proc. ACM MOBICOM*, pp. 1-9, 1998.

[25] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, 1997.

[26] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case", *IEEE Trans. on Networking*, vol. 1, no. 3, pp. 344-357, June 1993.

[27] K. M. Sivalingam, M. Srivastava, P. Agrawal, and J. C. Chen, "Low power access protocols based on scheduling for wireless and mobile ATM networks", *Proc. IEEE Universal Personal Communications Conference*, pp. 420-433, 1997.

[28] J. G. Proakis, *Digital Communications*. McGraw Hill, 1989.

[29] K. Cho and H. Samueli, "A 8.75-MBaud Single-Chip Digital QAM Modulator with Frequency-Agility and Beamforming Diversity", *Proc. IEEE Custom Integrated Circuits Conference*, pp. 27-30, 2000.

[30] PARSEC parallel simulation language (http://pcl.cs.ucla.edu/projects/parsec)

Power-Aware Acoustic Processing

April 25, 2003

Ronald Riley, Brian Schott, Joseph Czarnaski, and[*]
University of Southern California Information Sciences Institute
3811 N. Fairfax Dr., Suite 200, Arlington VA 22203-1707

Sohil B. Thakkar
University of Maryland, College Park

ABSTRACT

We investigated the tradeoffs between accuracy and battery-energy longevity of acoustic beamforming on disposable sensor nodes subject to varying key parameters: **1)** *number of microphones*, **2)** *duration of sampling*, **3)** *number of search angles*, and **4)** *CPU clock speed*. Beyond finding the most energy efficient implementation of the beamforming algorithm at a specified accuracy, we seek to enable application-level selection of accuracy based on the energy required to achieve this accuracy. Our energy measurements were taken on the HiDRA node, provided by Rockwell Science Center, employing a 133-MHz StrongARM processor. We compared the accuracy and energy of our time-domain beamformer to a Fourier-domain algorithm provided by the Army Research Laboratory (ARL). With statistically identical accuracy, we measured a *300x* improvement in energy efficiency of the CPU relative to this baseline. We also present other algorithms under development that combine results from multiple nodes to provide more accurate line-of-bearing estimates despite wind and target elevation.

1.   Introduction

Our objective is to develop software and hardware that support dynamic control of the use of energy and computational resources at various levels according to the need for precision and the readiness to expend

the required resources. We envision sensor nodes lying almost dormant for months before a sound emitting target comes within range. During this long quiescent period, the nodes must remain active enough to detect the incursion of an emitter and remain in contact with neighboring nodes to alert them of the detection. At this stage there is no need to determine the direction to an emitter with any accuracy, it is sufficient to detect it as a tripwire.
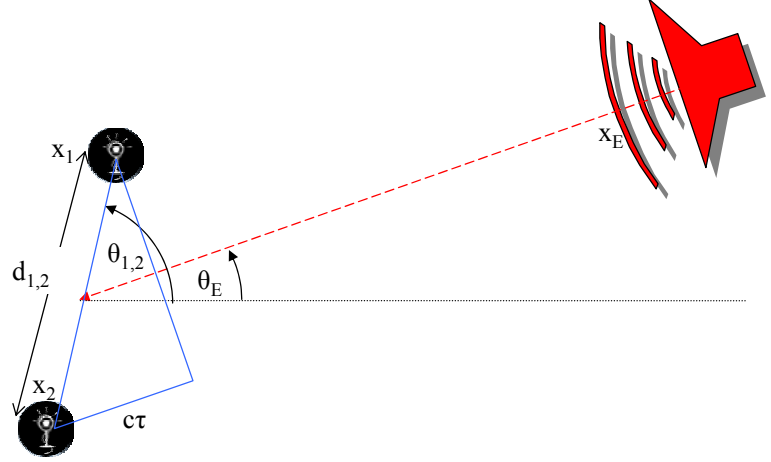
Figure 1. Beamforming Geometry.

Once an emitter has been detected, application level algorithms will determine how often the line of bearing (LOB), also referred to as the direction of arrival (DOA), is needed and the accuracy required to track and to identify the emitter. These decisions will depend on the perceived importance of the emitter and the amount of energy needed to provide the required accuracy. This paper summarizes our efforts to find the key parameters in acoustic beamforming and determine their relative contributions to the accuracy and energy consumption.

2.   Beamforming Algorithms

Acoustic beamforming algorithms estimate the LOB to distant acoustic emitters by time-shifting signals from microphones at known relative locations to form beams from selected directions. The two beamforming algorithms compared in this paper differ primarily in how they implement time-shifting the signals. The LOB can be thought of as a by-product of the optimal reconstruction of the signal from an emitter by shifting and adding the signals from a number of microphones[1].

As illustrated in Figure 1, sound from a distant emitter at position x_E arrives at two microphones at x_1 and x_2 with a relative delay

$$\tau_{1,2} = \frac{d_{1,2}}{c} \cos\left(\theta_E - \theta_{1,2}\right) = \left[d_{1,2}\cos\left(\theta_{1,2}\right)\right]\left\{\frac{\cos\left(\theta_E\right)}{c}\right\} - \left[d_{1,2}\sin\left(\theta_{1,2}\right)\right]\left\{\frac{\sin\left(\theta_E\right)}{c}\right\}, \tag{1}$$

where $d_{1,2} \equiv |x_2 - x_1|$ is the distance between the microphones, $\theta_{1,2}$ is the direction of the separation vector between the microphones, θ_E is the LOB of the emitter, and $c \sim 332\ m/s$ is the speed of sound in air

$$c_0 \approx \sqrt{T_{Kelvin}}\,(20.06\ \text{m/s}) = \sqrt{273.15 + T_{Celsius}}\,(20.06\ \text{m/s}). \tag{2}$$

Assuming the microphone geometry is either specified by rigid placement and/or calibrated after deployment, and the speed of sound is adjusted to the local climate, the LOB to the emitter, relative to the microphone separation vector, can be calculated from an estimate of this delay as

$$\theta_E - \theta_{1,2} = \pm \cos^{-1}\left(c\,\tau_{1,2}/d_{1,2}\right), \tag{3}$$

where the ambiguity in sign is due to taking the inverse of the cosine, an even function.

The delays can be estimated by minimizing the differences between signals S^1 and S^2 recorded at microphones 1 and 2,

$$S^1(t) = \alpha S^2(t - \tau_{1,2}),\qquad\qquad(4)$$

where α is the relative gain between the two signals. The search for the best delay is limited by (1) to $|\tau_{1,2}| < d_{1,2}/c$. Once the shift has been determined, the signals from the microphones can be shifted and added to provide an enhanced estimate of the signal from the emitter. The background noise and sounds from other emitters will be reduced in proportion to the number of microphones.

Methods for determining the LOB based on reconstructing the emitted signal, as in (4), from an acoustic array with M microphones can distinguish at most M-1 emitters. The additional uncertainty of the position of the emitters results in an underdetermined system of linear

Figure 2. Power of delay-summed at 18 search angles.

equations if we try to resolve M emitters. Resolving more emitters would require additional constraints such as knowing the power-spectrum of the emitters. Algorithms, such as MUSIC[2] and ESPRIT[3], which do not attempt to reconstruct the emitted signals can resolve, at most, M emitters from an array of M microphones. These algorithms perform eigenvector decomposition of the *MxM* covariance matrix of the signals from the M microphones and so do not lend themselves to efficient implementation in integer-math oriented processors in sensor nodes.

Beamforming is accomplished by time-shifting the signals from an array of microphones to a common position with time delays prescribed by (1), for an assumed LOB. Beams are formed for a number of LOB search angles. As the search angle approaches the correct LOB of the emitter, the signals constructively interfere as in (4). As shown in Figure 2, the search angle with the maximum power in the delay-summed signal is selected as the estimated LOB. This LOB estimate is refined by parabolic interpolation of the power over the adjacent search angles.

The two algorithms compared in this paper differ primarily in how they shift the signals. The baseline algorithm from ARL performs a floating-point FFT on each signal then shifts, sums, and computes the beam power in the Fourier domain. Our beamformer performs all of these operations in the time domain in integer math.

Our beamformer assumes a single target and employs common integer code to report the LOB as the direction with the delay-summed combined signal of loudest total acoustic energy. These results do not include code for false alarm rejection such as ARL's Harmonic Line Analysis. We plan to incorporate these capabilities into future implementations.

Our algorithm is roughly **300x** more efficient in the consumption of energy by the CPU than the baseline. The largest contribution to this efficiency (~**20x**) is due to <u>floating-point</u> emulation on the StrongARM processor. The next most significant contribution (~**10x**) is our ability to use <u>fewer samples</u> to achieve similar accuracy. Fast Fourier transforms operate on vectors containing integer powers of 2 samples ($S=2^n$). We did not modify the baseline algorithm to vary the number of samples from its hard-coded value of $S=1024$. Operating in the <u>time domain</u> provided the final contribution of (~**3/2x**,) by not Fourier transforming the signals.
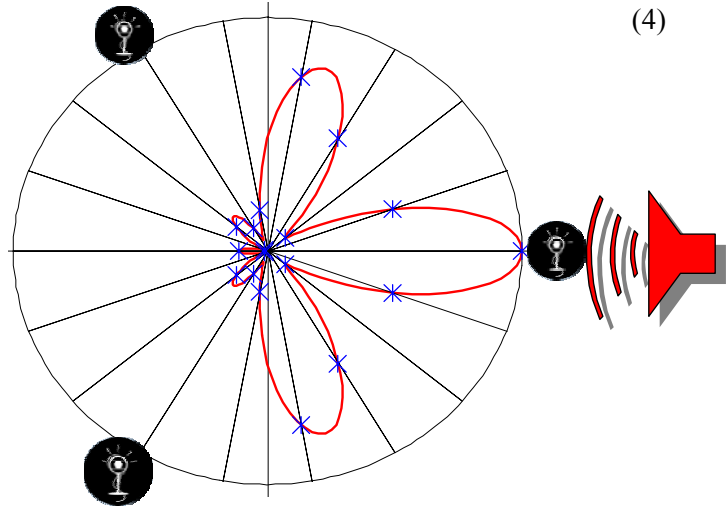
3. LOB Beyond Beamforming

The algorithms discussed in this section are under development and optimization for sensor nodes. We anticipate that these will lead to significant improvements over beamforming in accuracy/noise insensitivity, false-alarm rejection, and an extended range in tradeoffs between accuracy and energy.

Beamforming makes a number of limiting assumptions 1) that the microphones are fixed on a rigid plane, 2) that the emitter lies on the same plane, and 3) that there is little or no

Figure 3. Uncertainty in LOB of tracked vehicle.

wind. While beamforming uses (1) to construct a set of delays corresponding to various search angles and selects the "best" angle, the LOB can also be estimated directly from (1) based on estimates of the time shifts between pairs of microphones. Given a set of *M>2* microphones with synchronized sampling, we can select up to *M! / [2 (M-2)!]* pairs. For example, three microphones provide three pairs. Assuming that the geometry of the *M* microphones has been calibrated, (1) provides a system of *M! / [2 (M-2)!]* equations in two unknowns, the sine and cosine of the LOB divided by the speed of sound. For more than two pairs, the over-determined system of equations can be solved by standard weighted least-squares methods. Increasing the number of microphones and pairs should improve the accuracy.

The LOB and the speed of sound can be extracted from the results in integer math with the CORDIC[4] as

$$\theta_E = \tan^{-1}\left[\left\{\frac{\cos(\theta_E)}{c}\right\}, \left\{\frac{\sin(\theta_E)}{c}\right\}\right], \tag{5}$$

and

$$c = \frac{1}{\sqrt{\left\{\frac{\cos(\theta_E)}{c}\right\}^2 + \left\{\frac{\sin(\theta_E)}{c}\right\}^2}}. \tag{6}$$

Although, it may appear that the speed of sound should be constrained to the constant predicted in (2) there are cases in which the apparent speed of sound can vary. When noise results in nonphysical estimates of the apparent speed of sound, the LOB estimated in (5) can be iteratively refined based on small angle corrections with the speed of sound held to the predicted constant in (1) as

$$\left\{d_{1,2}\sin(\theta_E - \theta_{1,2})\right\}\delta\theta_E = c\tau_{1,2} - \left\{d_{1,2}\cos(\theta_E - \theta_{1,2})\right\}. \tag{7}$$

The RMS error of the system of equations defined by (7), or (1), provides a measure of the angular uncertainty of the LOB estimate. If some pairs of microphones produce multiple delays due to multiple targets, this error and the apparent speed of sound provided by (6) could be used to select delays into sets for each target.

Figure 3 shows a plot of this uncertainty for a tracked vehicle driving around an oval track. The vehicle is about 600 meters from the sensor node at an LOB of 280° counterclockwise from east (roughly south). This is based on acoustic data collected by USC-ISI at ARL's Aberdeen Proving grounds with HiDRA nodes, described in section 5, using with three microphones deployed on an equilateral triangle with 7-foot separation.
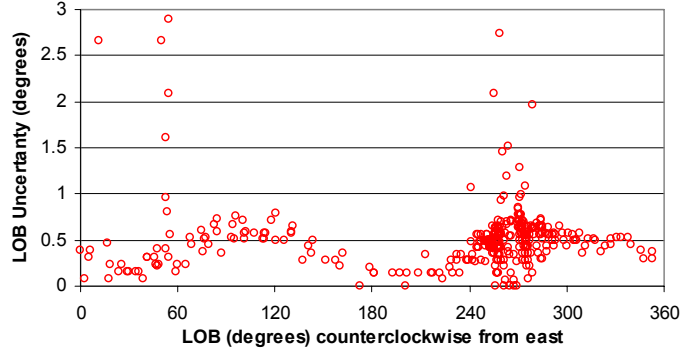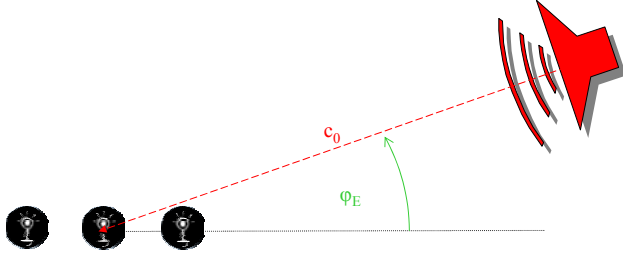
Figure 4. Elevation angle of emitter.
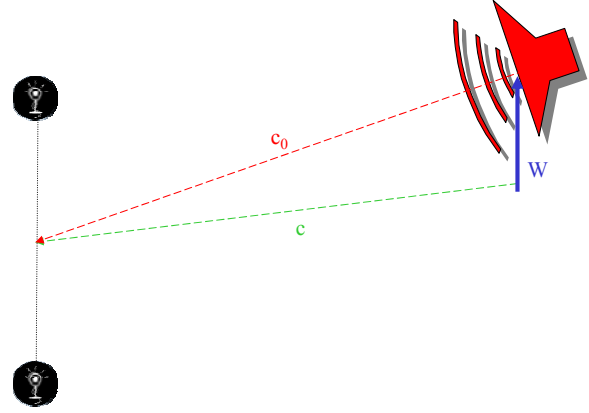
Figure 5. Wind speed geometry.

3.1.   Emitter Elevation

If the emitter is elevated above the plane on which the microphones are distributed, as shown in Figure 4, this decreases the delays between signals and increases the apparent speed of sound estimated by (6) as

$$c = \frac{c_0}{\cos(\phi_E)},$$ (8)

where φ_E is the elevation angle of the emitter. As the emitter moves directly above the microphones, the apparent speed would tend toward infinity as all of the time shifts approach zero. The elevation angle of the emitter could be estimated by generalizing our analysis to 3D and elevating one or more microphones above a plane containing three or more microphones[5]. If we can segregate the contribution due to elevation from other effects, such as wind, by collaboration between sensor nodes, each with three or more microphones, we could estimate the elevation angle from (8).

3.2.   Wind-Speed Vector Effects

As illustrated in Figure 5, the average wind speed W between the emitter and the microphones will add as a vector to the sound radiating from the emitter at speed c_0 resulting in an apparent speed of sound of

$$c = \sqrt{\left| c_0 + W \cos(\theta_E - \theta_W)\right|^2 + W^2 \sin^2(\theta_E - \theta_W)},$$ (9)

where θ_W is the direction that the wind is coming from. The estimated LOB would also be shifted by

$$\delta\theta_E = \tan^{-1}\left[c_0 + W \cos(\theta_E - \theta_W), W \sin(\theta_E - \theta_W)\right].$$ (10)

Assuming that the wind speed is small compared to that of sound ($W << c_0$,) we can approximate

$$(c - c_0) \sim W \cos(\theta_E - \theta_W) = \cos(\theta_E)\{W \cos(\theta_W)\} + \sin(\theta_E)\{W \sin(\theta_W)\},$$ (11)

and

$$\delta\theta_E \sim \frac{(c - c_0)}{c}\tan(\theta_E - \theta_W).$$ (12)

A wind speed of 14 mph orthogonal to the LOB would result in an LOB error of about 1 degree.

The difference of apparent from expected sound speed is plotted in Figure 6 for a tracked vehicle driven on an oval track around the sensor node for a number of orbits. These preliminary estimates of the apparent sound speed are based on delays of maximum correlation (15), described in the next section.

155
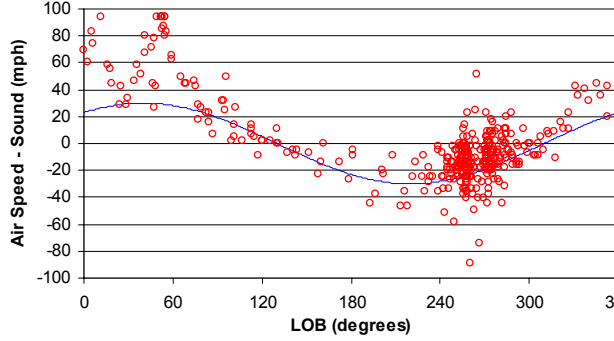
Figure 6. Deviation of sound speed vs. angle.   **Figure 7. Power spectrum of tracked vehicle.**
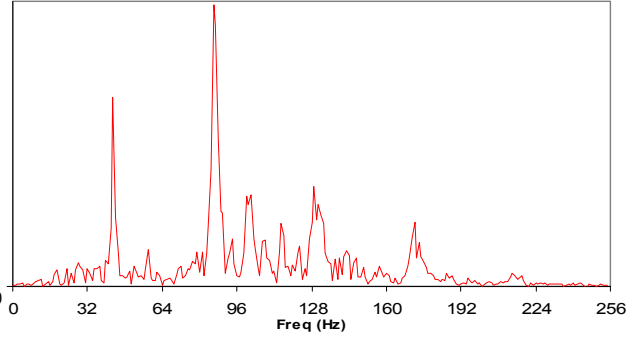
These estimates are compared to the expected deviation based on (11) for a wind speed of 30 mph. This is somewhat larger than the 10-mph average wind speed measured during data collection. The wind direction, taken from the ground truth, agrees well with the estimated deviations in sound speed.

Given an estimate of the wind direction, we can estimate and correct the offset in the LOB from (12) based on the apparent speed of sound, if we can assume that there are no other contributions such as elevation of the emitter above the sensor plane.

If we combine the apparent sound speed and estimated LOB from two or more sensor nodes with enough separation transverse to the LOB of the emitter such that they provide significantly different estimated LOBs, and assume that the wind is roughly the same for all sensor nodes, we can solve the resulting system of linear equations given by (11) for the wind vector components. The wind direction can be computed using the CORDIC algorithm as

$$\theta_W = \tan^{-1}[W\cos(\theta_W), W\sin(\theta_W)], \tag{13}$$

and the result can be used in (12) to infer the offset of the LOB's due to the wind.

The LOB and apparent speed of sound from three or more nodes can be combined to solve for the wind and elevation simultaneously by combining (8) and (11) as

$$c \sim \left\{\frac{c_0}{\cos(\varphi_E)}\right\} + \cos(\theta_E)\{W\cos(\theta_W)\} + \sin(\theta_E)\{W\sin(\theta_W)\}. \tag{14}$$

This assumes that all of the nodes have the same emitter elevation, which could be satisfied if all of the nodes are at the same elevation and roughly the same distance from the emitter. We would have to constrain the added unknown such that the inferred cosine is in the assumed range (0, 1) and would interpret the elevation angle to be positive, forcing the object to lie above the sensor plane.

3.3. Signal-Pair Delay Estimation

All of this analysis is based on the ability to estimate the delay between signals collected by a pair of microphones. One estimate of the delays can be obtained by finding the shift between the signals that produces a peak in correlation between the signals from microphone 1 (S^1) and 2 (S^2)

$$C^{1,2}(\tau_{1,2}) = \sum_t S^1(t)S^2(t - \tau_{1,2}). \tag{15}$$

This estimate is best suited to emissions of short duration, such as gunfire or explosions, and is sufficient for cases with high signal to noise and broad band signals.

Much of the recent work in noise-tolerant LOB estimates are based on the fact that many targets of interest, such as ground and air vehicles, emit sound in a number of narrow acoustic bands[6], as shown by the acoustic power spectrum of a tracked vehicle shown in Figure 7. If multiple targets emit primarily in different narrow frequency ranges, their LOBs could be determined by a single sensor node with as few as three microphones.

The LOB of frequencies from 32 to 128 Hz is plotted in Figure 8 for a sensor node with three microphones and two tracked vehicles. The average power spectrum of the three microphones is also plotted. The LOBs

Figure 8. LOB of spectral components.

cluster around 240° and 77° corresponding to the two vehicles. The LOB is plotted as zero for frequencies where it is undefined. Once the LOBs of the two vehicles are estimated, it is straight forward to separate their line-spectra as a basis for target classification.

For each spectral component, the LOB of is based on time shifts of microphone pairs estimated as

$$\tau_{1,2}(\omega) = \frac{\{\varphi[S^1(\omega)] - \varphi[S^2(\omega)]\} + 2n\pi}{\omega}, \tag{16}$$

where the phase of the Fourier component of the signal $\varphi[S^l(\omega)]$ can be computed with the CORDIC algorithm and the time shift is limited to the range $[-d_{1,2}/c, d_{1,2}/c]$. For higher frequencies ($f>c/(2d_{1,2})$), there can be more than one solution to (16). This degeneracy can be resolved by selecting the solution that is most consistent with lower frequency components or is most consistent with the expected speed of sound and error of fit based on the solution to (1). This ambiguity can be avoided by restricting microphone separations to $d<c/(2f)$, where f is the highest frequency of interest in the emitters spectrum.

Our current implementation of the acoustic beamformer operates in the time rather than the frequency domain in order to provide a marginal (*3/2x*) increase in power efficiency. By using an integer FFT we could achieve comparable efficiency and add an adjustable parameter corresponding to the sampling rate of the acoustic data. Although the sampling rate of the digitizer will probably be fixed at ~1024 Hz, the captured signal can be downsampled to provide significant reductions in energy requirements with little effect on the accuracy of the LOB.

4.   Algorithmic "Knobs"

The accuracy and energy requirements of acoustic beamforming depend on a number of parameters, as suggested by Figure 9. Some are largely dictated by the application. We have chosen as our set of independent variables: **1)** number of microphones *M*, **2)** number of acoustic samples *S*, **3)** number of beams *B*, and **4)** CPU clock speed f_{CPU}.

4.1.   Number of Microphones (M)

The number and placement of microphones is principally a hardware design parameter but can also be used as an algorithmic parameter by selecting a subset of the signals collected. We will show that the
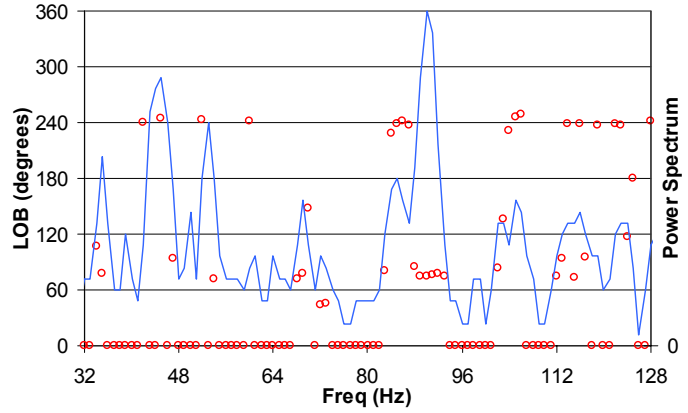
system energy required for beamforming is *proportional* to the number of microphones, but there are negligible improvements in accuracy due to adding more microphones at the same radius.

We limit our analysis to nodes supporting at least two microphones. Although single-microphone nodes could collaborate to determine the direction to an emitter, we exclude this case due to the requirements of a

Figure 9. Beamforming Algorithmic Knobs.

large amount of energy to move raw data between nodes, of 10-us synchronization between nodes, and of 5-mm accuracy in relative positioning of microphones. Although the ambiguities resulting from two-microphone beamforming would require collaboration, only a trivial amount of data need be exchanged. Synchronization to a fraction of a second is sufficient, and the required accuracy of relative positioning is similarly relaxed.

4.2. Number of Acoustic Samples (S)

For a fixed sampling rate, in our case 1024 Hz, the system energy required for beamforming is *proportional* to the number of samples simultaneously collected from each microphone. The number of samples collected per second for each microphone is typically selected to be 1 kHz for acoustic tracking of vehicles to facilitate beamforming with the spectrum above 250 Hz attenuated to filter out wind noise. We implemented this as analog anti-aliasing filtering.

4.3. Number of Beams (B)

The beam power is computed at a number of evenly spaced search angles, and interpolated by a parabolic fit to estimate the angle with maximum power. The number of beams only affects the execution speed of the algorithm, *not data acquisition*. The system energy for beamforming is linear with, not proportional to, the number of beams searched.

4.4. CPU Clock Speed (f$_{CPU}$)

The Highly Deployable Remote Access Network Sensors & Systems (HiDRA) provided by Rockwell Science Center (RSC)[7,8,9], supports software-control of the clock speed of the StrongARM CPU over the range 59-133 MHz. Unfortunately, voltage scaling is not supported in this hardware. Without voltage scaling, clock scaling may appear to only change how long the algorithms run without changing the energy requirements. However, the power consumed by other components in the system during execution of the algorithm and the power consumed by the CPU during data collection give this parameter utility in reducing system energy required for beamforming.

5. System Power Equation

The energy consumed by the HiDRA node to capture data and compute the LOB is modeled by

$$E = \frac{S}{f_S} P_{AD} + \left[\frac{S}{f_S} + \frac{f_{133}}{f_{CPU}} T_{A\lg} \right] \left[P_{Ovr} + \frac{f_{CPU}}{f_{133}} P_{CPU} \right], \tag{17}$$

where f_{CPU} is the clock rate of the StrongARM CPU and f_{133} = 133 MHz is the maximum clock rate.

This equation reflects that the power required by the A/D board, P_{AD}, is only consumed during the data acquisition time which is the number of samples acquired, S, divided by the sampling rate f_S=1024 Hz. Also implicit in this equation is that, for the HiDRA node, the CPU and memory consume power, P_{CPU}, while the samples are acquired, but the algorithm does not begin to run, for a period T_{Alg}, until after data collection is complete. There is also a significant overhead power, P_{Ovr}, consumed by other supporting components such as the radio.

The CPU power and execution time of the algorithm are proportional and inversely proportional, respectively, to the CPU clock rate, f_{CPU}. The clock rate resulting in the minimal energy consumption can be found by setting the derivative of (17) with respect to f_{CPU} to zero, resulting in

Figure 10. RSC HiDRA sensor node stack.

$$f_{CPU} = f_{133} \sqrt{P_{Ovr} T_{Alg} / T_{AD} P_{CPU}} \ , \tag{18}$$

where T_{Alg} is the execution time of the algorithm and P_{CPU} is the CPU power at the CPU clock rate of 133 MHz.

When the CPU and overhead power components are roughly equal and the execution time of the algorithm at 133 MHz is a small fraction of the sampling window, as with our algorithm, the minimum available CPU clock rate of 59 MHz provides the best energy efficiency. When the algorithm runs longer than the sampling window, as with the ARL algorithm, the maximum available CPU clock rate of 133 MHz gives the best results.

6. Power Measurements on the HiDRA Node

The power measurement test setup consisted of the HiDRA sensor node and a current sensing resistor placed inline with the input voltage connector. The voltage drop across this resistor was measured using a National Instruments PCI-MIO-16E-1 data acquisition card and triggered from GPIO lines on the node's processor module. The HiDRA node, as shown in Figure 10, contains the following modules:

1) Processor/Memory: SA1100 processor board running eCos at 59-133 MHz., with 4 MB of ROM (flash memory), 1 MB RAM, input voltage 3.3V I/O, and 1.5V core.

2) DC/DC: input voltage from 2 9-V batteries or DC adapter. Our test measurements were taken with a 12-V DC input. This module supplies voltages 3.3V and 1.5V for the I/O lines and processor core respectively and separate analog voltage lines for the A/D and Radio modules.

3) Radio: A 900 MHz Rockwell proprietary radio.

4) A/D: 5 channels, 3 Multiplexed with variable gains of 1x, 2x, 5.02x, 10.09x, 20.12x and 2 individual inputs with variable gains of 10x, 43.32x, 30x, 36.68x, 49.98x. The selectable gains are tuned to specific sensors that RSC uses for this platform. The acoustic data was captured using only the multiplexed sensor input to keep the gains equivalent across all channels. Low-pass anti-aliasing filters with a cutoff
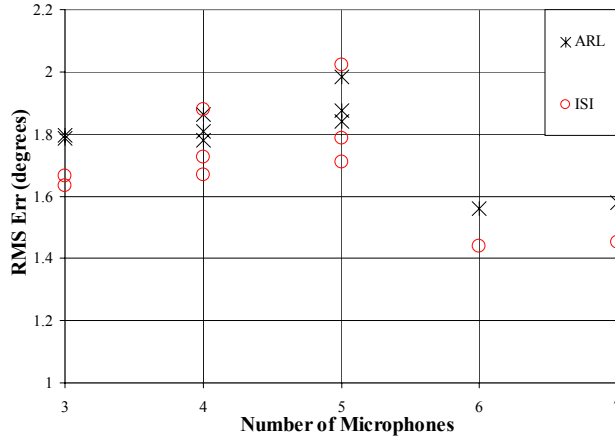
Figure 11. Beamforming errors of algorithms for 1024 samples, 20 beams, and selected subsets of the seven microphones.
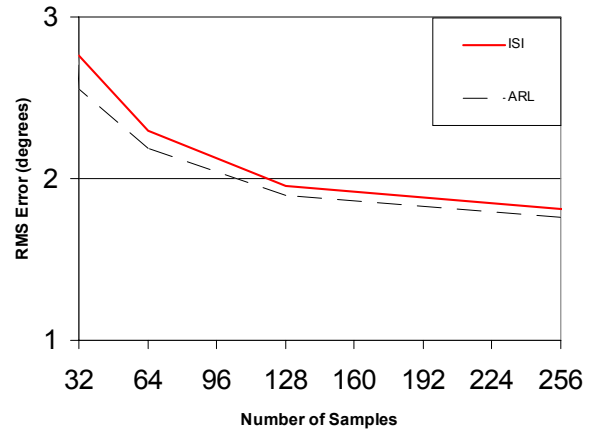
Figure 12. Beamforming errors of algorithms for seven microphones and 20 beams vs. number of samples used.

frequency of 3 kHz were also added to the inputs of each of these channels to eliminate cross talk between the respective channels.

The current measurements were taken at the full clock rate of 133 MHz, which is the optimal operating frequency for the HiDRA node, and at various increments down to 59 MHz. Other power measurements were taken for the different operating modes required to form a single LOB. These modes include acquiring the data from the input microphones, executing the algorithms, and putting the processor in sleep mode. The individual power consumption of each module was measured by removing boards from the stack and calculating the difference in current through the sensing resistor.

7.   Results

We evaluated the two algorithms on a test set of acoustic data collected by ARL collected at their Aberdeen Proving Grounds. The data was synchronously collected from six microphones uniformly distributed on a 4-foot radius circle, and a seventh microphone in the center. A single military vehicle was driven by this acoustic array with GPS to provide ground-truth position at 1-second intervals. The signals were collected continuously on a seven-channel digitizer at 12-bits per channel at 1024 samples per second.

For our tests, we broke the acoustic data up into 1-second records associated with the available position ground truth. For each 1-second record, we computed the LOB with both algorithms and measured their errors relative to the ground truth. The results presented are of the Root-Mean-Square of these errors over all of the records in the set.

In addition to validating that our algorithm provided results statistically identical to those of the baseline, we investigated the effect of the various knobs on the accuracy of both algorithms.

As illustrated in Figure 11, increasing the number of microphones on a circle above the minimum of three provides little, if any, improvement in the accuracy of the LOB estimate. Doubling the number of microphones from 3 to 6 reduces the RMS error only by 12% for this single emitter.From the errors plotted in Figure 12, we can see that 32 is the minimum number of samples for which the algorithms can

provide a useful result. However, using more than 128 samples provides only modest improvements in accuracy.

As shown in Figure 13, the error grows rapidly for fewer than eight beams, but improves only slightly for more than sixteen beams. These results depend on the dominant frequency of the acoustic spectra emitted by the target and the "knee" of the curve is likely to shift for other vehicles.

We optimized the execution time of our code using the web-based JouleTrack[10] emulator before porting it to the HiDRA. Although

JouleTrack did not provide identical results to what we measured directly from the hardware, it was useful guide in modifying the code to reduce execution time and energy use.

We electronically measured the power consumed by each component of the node during various modes of operation and we measured the execution time of the two algorithms at the maximum and minimum clock rates, as shown in Figure 14. The results of these measurements are summarized in Table 1.

Assuming three microphones (M=3), twelve beams (B=12), and a full second of data (S=1024), applying the values in this table to (17) results in a total system energy of 1614 mJ for the baseline algorithm and 746 mJ for the ISI algorithm, as shown in Figure 15. Despite a 250x reduction in execution time for the same clock rate, our algorithm results in a modest 2x reduction in overall system energy required to acquire the data and process it to produce a single LOB.

Under similar conditions, but with only 0.125 sec sampling window (S=128), the system energy for the baseline would be 816 mJ and the system energy for the ISI algorithm
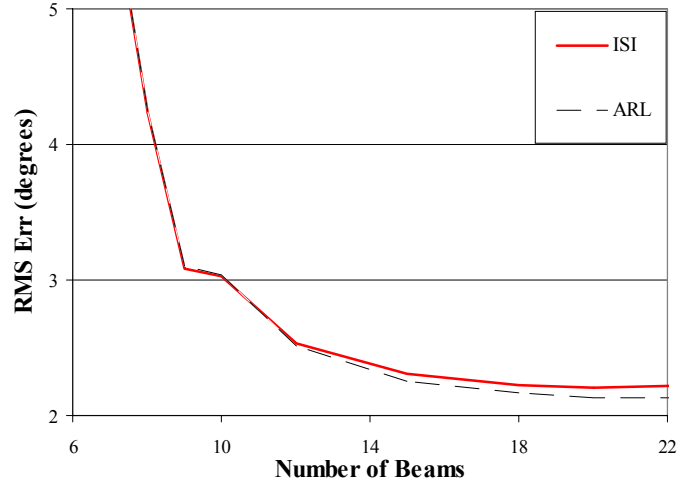
Figure 13. Beamforming errors for 64 samples and seven microphones vs. number of beams.
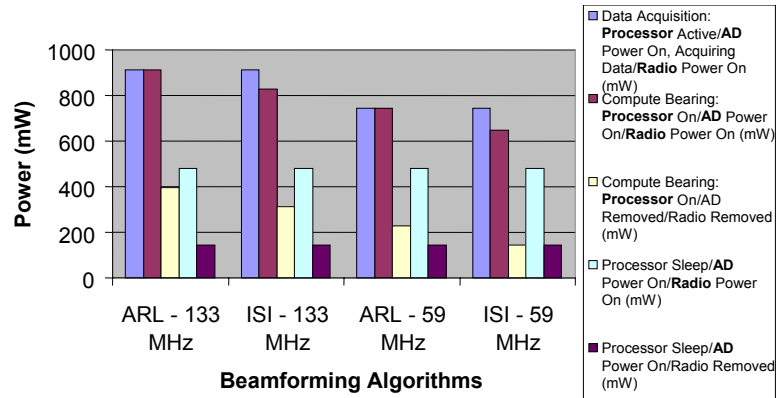
Figure 14. Power Consumption of Various Node Modes.
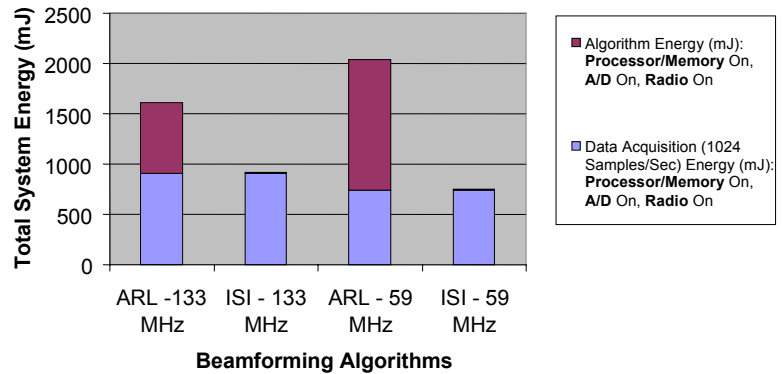
Figure 15. Acquisition/Proc. Energy to Compute a LOB.

would be 93 mJ. *Our algorithm would provide a 9x reduction in system energy over the baseline algorithm under these conditions.*

Table 1. Power Measurements for the HiDRA node.

	P_{Ovr} mW	P_{AD} mW	f_{CPU} MHz	P_{CPU} f_{CPU}/f_{133} mW	T_{Alg} f_{133}/f_{CPU} μs
ISI	274	252	59	132	M B S 0.0555
ARL	358	252	133	302	M B 14300

8. Conclusions

We have demonstrated the ability to perform acoustic beamforming over a range of accuracy requiring a corresponding range of energy by introducing a number of "knobs" into the basic algorithm. This analysis led us to develop an algorithm that is roughly *300x* more efficient than the baseline in CPU energy. More importantly, our algorithm provides *9x overall system energy savings* when LOB estimates of lower accuracy are sufficient.

We found that adjusting the number of microphones *M* offered the largest improvements to energy efficiency with the least impact on accuracy of the LOB. However, this parameter can only be effectively adjusted during the design of the sensor node. It is difficult to construct a board capable of collecting a large number of synchronized signals but only capture a selected subset without paying much of the power penalty of sampling all of the signals. We determined that three microphones are the minimum required to perform beamforming without collaboration with other nodes.

The number of samples collected, *S,* for each microphone offers comparable improvements to system energy efficiency with slightly larger impact on accuracy. However, this parameter can be adjusted over a wide range, 32-1024, resulting in a *16x range of system energy requirements*. The memory requirements of future nodes could be reduced by reducing the sampling window and by using our integer algorithm rather than the floating-point baseline.

The number of beams formed in the software, *B,* only modulates the algorithm execution time which is only a fraction of the sampling window. So this parameter has only a minor effect on the energy efficiency of the whole system.

The clock speed of a CPU, *f_{CPU}*, typically has a limited range of values, in the case of the HiDRA node 59-133 MHz. We have shown that a faster than real-time algorithm will be most efficient at the lowest available clock speed. This will be even more the case for more power-aware sensor nodes being designed that will be capable of voltage scaling and of computing the algorithm during data collection.

9. Future Work

We plan to apply what we have learned from the HiDRA node to developing a more power-aware sensor node under the PASTA contract at USC-ISI funded by DARPA. The PASTA architecture will represent a different approach to a sensor node design. Instead of a processor-centric design, where the main CPU controls all functionality of each module in the stack, the modules will act as independent units in a microcontroller network. Each module will have a low power *8051* based microcontroller operating over an *I2C* control network operating at *100 kbs*. This controller will provide the main communication between modules for power control and module to module communication. There will also be a higher speed data channel between modules for transmissions that require higher data rates. This network will allow each module to perform its required task and, when it is not needed, it will either shutdown or go into a low power sleep condition. Modules that have a higher quiescent-current consumption in their idle

modes can be removed from the overall power budget if they are not need to perform a required task. An example would be performing data acquisition and computation on a low-power DSP and then sending the results to a radio module over the microcontroller network. The main processor will not be used in this mode and thus will either be turned off or put in a sleep cycle.

Each micro-sensor will have interchangeable hardware with mission specific processing modules, distributed power monitoring, voltage scaling, and sleep management. It will also incorporate techniques and technologies developed under the PAC/C Phase I program (PADS) to include deep-submicron techniques to reduce idle-mode power leakage in digital circuits with minimal or no degradation in performance.

Figure 16. Fabric Beamformer.

We have collected three-channel raw acoustic data with the HiDRA nodes and we have ported the beamforming codes to these and a variety of sensor nodes. During our next data collect, we plan to have the beamforming code providing realtime LOB results on sensor nodes in the field. The sensor nodes fielded with live acoustic beamforming may include HiDRA, PASTA, and the prototype fabric beamformer[11] shown in Figure 16, developed by USC-ISI and Virginia Tech. under the STRETCH e-textile effort funded by DARPA.

10.   References

[1] D. K. Campbell, Adaptive Beamforming Using a Microphone Array for Hands-Free Telephony. Masters Thesis, Virginia Polytechnic Institute and State University, 1999. Available: http://scholar.lib.vt.edu/theses/available/etd-022099-122645/

[2] R. Schmidt, "Multiple emitter location and signal parameter estimation," IEEE Trans. Antennas Propagat., vol. AP-34, pp. 276–280, 1986.

[3] R. Roy and T. Kailath: "ESPRIT—Estimation of Signal Parameters via Rotational Invariance Techniques," IEEE Trans. Accoust. Speech and Signal Proc., vol. 37, no. 7, pp.984–995, July 1989.

[4] Ray Andraka, "A survey of CORDIC algorithms for FPGAs (121K),"FPGA '98. Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, Feb. 22-24, 1998, Monterey, CA. pp191-200. Available: http://www.fpga-guru.com/cordic.htm

[5] M. Walworth and A. Mahajan, "3D Position Sensing Using The Difference In The Time-Of-Flights From A Wave Source To Various Receivers," ICAR '97, Monterey, Ca, July 1997.

[6] Tien Pham; Sadler, B.M., "Adaptive wideband aeroacoustic array processing," Statistical Signal and Array Processing, 1996. Proceedings., 8th IEEE Signal Processing Workshop on (Cat. No.96TB10004 , 1996 Page(s): 295 -298. Available: http://eewww.eng.ohio-state.edu/~randy/Microphone_Reading_List/index.html

[7] J. Agre, L. Clare, G. Pottie and N. Romanov, "Development Platform for Self-Organizing Wireless Sensor Networks," Proceedings of the SPIE Conference on Unattended Ground Sensor Technologies and Applications, Orlando, FL, April, 1999, SPIE Vol. 3713, pp. 257-268.

[8] "Highly Deployable Remote Access Network Sensors & Systems", Rockwell Science Center. Available: http://www.rsc.rockwell.com/hidra

[9] J. Montague and Control Engineering staff, "Progressive Innovations", Control Engineering, March 1, 2002. Available: http://www.controleng.com/

[10] A. Sinha and A. Chandrakasan. "JouleTrack - A Web Based Tool For Software Energy Profiling." Proc. 38th Design Automation Conference, June 2001. Available: http://dry-martini.mit.edu/JouleTrack/

[11] Edmison, J., Jones, M., Nakad, Z., and Martin, T. "Using Piezoelectric Materials for Wearable Electronic Textiles," Proceedings of the Sixth International Symposium on Wearable Computers, Oct. 2002, to appear.

Power Management for Energy-Aware Communication Systems

CURT SCHURGERS, VIJAY RAGHUNATHAN, and MANI B. SRIVASTAVA
University of California, Los Angeles

System-level power management has become a key technique to render modern wireless communication devices economically viable. Despite their relatively large impact on the system energy consumption, power management for radios has been limited to shutdown-based schemes, while processors have benefited from superior techniques based on dynamic voltage scaling (DVS). However, similar scaling approaches that trade-off energy versus performance are also available for radios. To utilize these in radio power management, existing packet scheduling policies have to be thoroughly rethought to make them energy-aware, essentially opening a whole new set of challenges the same way the introduction of DVS did to CPU task scheduling. We use one specific scaling technique, dynamic modulation scaling (DMS), as a vehicle to outline these challenges, and to introduce the intricacies caused by the nonpreemptive nature of packet scheduling and the time-varying wireless channel.

Categories and Subject Descriptors: H.1.1 [**Models and Principles**]: Systems and Information Theory—*General systems theory*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms: Algorithms, Performance, Design

Additional Key Words and Phrases: Energy-efficient, wireless communications, adaptive, scaling

1. INTRODUCTION

1.1 System Power Management

In commercial and research efforts alike, recent trends have shifted the emphasis in system design away from ever-increasing throughput alone. Instead, energy and power consumption are becoming ever more formidable and important constraints to address [Benini and de Micheli 1997; Pedram 2001]. Indeed,

technology integration and miniaturization have intensified cooling and packaging challenges. In addition, the energy supply itself is often limited to build-in batteries, as untethered operation is desired for mobility and portability. A class of systems where this observation holds especially true is that of personal communication devices, which are also becoming increasingly prevalent.

To reduce power spending without sacrificing performance, system designers have drawn on the concept of "energy or power awareness." The key idea is that the system only delivers the performance that is strictly required, thereby avoiding superfluous power consumption. Power and energy awareness are therefore often paraphrased as "the right power/energy at the right time and the right place." When designing according to this concept, there are two aspects to consider: the technique itself that introduces the awareness, and the power management strategy that exploits it.

The oldest and most straightforward technique is to shut down unused parts of the system [Benini et al. 1999]. Shutdown-based power management has been explored for hard disks, displays [Lorch and Smith 1998], and communication modules [Wang and Mandayam 2001], among others. For processors, it has been incorporated in the kernel of real-time operating systems (RTOS) [Srivastava et al. 1996]. However, a breakthrough came with the development of dynamic voltage scaling (DVS), a technique for digital circuits that is more effective than shutdown [Chandrakasan et al. 1992]. The reason is the convex nature of the power–speed curve, which comes about by varying the operating voltage. Numerous DVS-based power management strategies have been proposed to harness this potential [Burd et al. 2000; Govil et al. 1995; Gruian 2001; Gutnik and Chandrakasan 1997; Krishna and Lee 2000; Manzak and Chakrabarty 2000; Nielsen et al. 1994; Shin and Choi 1999; Weiser et al. 1994; Yao et al. 1995].

Unfortunately, energy awareness brought forth by DVS is restricted to digital circuits, such as processors, which can leverage DVS in their task-scheduling engine. Voltage scaling not being applicable; is it nevertheless possible to find a similar technique for other parts of the system? Or is shutdown the best we can do?

1.2 Power Management in Communication Subsystems

The search for superior power management techniques is especially relevant for the communication subsystem, and in this paper, we introduce and discuss a powerful scaling-based approach that can be viewed as the counterpart of DVS.

The importance of radio power management arises from the fact that communication is often the dominant power hog in the complete system [Raghunathan et al. 2002]. While the inherent energy consumption of digital circuits is rapidly decreasing due to Moore's law and ingenious design techniques, the power that is radiated to carry information does not follow this trend. This observation is particularly true for wireless RF (radio-frequency) devices, where the radiated power depends on the transmit distance and is physically constrained by Maxwell's laws. With the increasing proliferation

of cellular phones with Internet browsing capabilities, Bluetooth-connected PDAs, and wireless LANs, radio power management will undoubtedly gain in importance.

Radios typically posses a number of control knobs, which gives rise to a convex power-performance curve. As such, these knobs can be used to introduce energy awareness in a similar fashion as the operating voltage in DVS. Possible radio control knobs are the modulation level, the error coding, or combinations, or both. We refer to the resulting techniques as dynamic modulation scaling (DMS), dynamic code scaling (DCS), and dynamic modulation-code scaling (DMCS). To provide energy awareness, they need to be integrated into the system power management.

The aim of this paper is to provide insight into such scaling-based radio power management strategies, the associated challenges, and possible solutions. Just as people created energy-aware versions of RTOS task-scheduling policies, we investigate energy-aware packet scheduling as part of these power management strategies. As a vehicle for this exploration, we focus on dynamic modulation scaling (DMS), which we first introduced in Schurgers et al. [2001a]. It is a readily accessible control knob in a number of existing communication systems.

2. RELATED WORK

DMS for radios can be viewed as the counterpart of DVS for digital circuits, as both exploit the presence of a convex power–speed curve via the notion of scaling. We can therefore find inspiration in the way task scheduling was extended to make it energy-aware, when developing energy-aware packet-scheduling policies. Nevertheless, a direct correspondence is impossible due to the inherent differences between radios and digital circuits, which we detail in Section 3.4. Voltage scaling was first included in self-timed circuits [Nielsen et al. 1994] and later extended toward synchronous ones [Gutnik and Chandrakasan 1997], where a buffer is used to smooth the load and steer the adaptation. The initial DVS work in operation system research was in the context of a workstation like environment [Govil et al. 1995; Weiser et al. 1994], with average throughput as the performance metric. Task scheduling under real-time constraints, that is, the presence of deadlines, is considered in Burd et al. [2000], Gruian [2001], Krishna and Lee [2000], Manzak and Chakrabarty [2000], Shin and Choi [1999], and Yao et al. [1995].

In the realm of communications, a shutdown approach is proposed in Wang and Mandayam [2001], leveraging the time-varying nature of the wireless channel. Transmissions are deferred to times where the channel can support energy-efficient data transmission, while taking into account various delay constraints. A scheduling technique based on code scaling (DCS) is described in Prabhakar et al. [2001]. It finds an energy-efficient schedule for a set of packets that have to be transmitted before an overall deadline, and is similar to the work on processor scheduling presented in Yao et al. [1995]. The basic concept of modulation scaling (DMS) was first proposed by Schurgers et al. [2001a]. Here, we build upon this concept to describe a complete framework for dynamic power

management of the radio communication subsystem in wireless embedded systems.

We explore radio power management through DMS, which essentially relies on the ability of the radio to change its modulation on the fly. This is indeed practically feasible, and appropriate hardware architectures have been developed [Cho and Samueli 2000]. Originally, these architectures were used for a different purpose, namely adapting the modulation to maximize the system throughput [Balachandran et al. 1999; Ue et al. 1998; Webb and Steele 1995]. In this case, the appropriate choice of the modulation level only depends on the current condition of the wireless channel, and does not involve any scheduling decisions. On the other hand, as we illustrate in this paper, energy awareness is strongly related to scheduling, where current decisions and future events are tightly interwoven. Although utilizing the same radio control knob, DMS and throughput maximization not only serve a different purpose, but also require completely different adaptation policies. This change in mindset is similar to the one that occurred in the CPU world when the design goal was switched from throughput maximization to energy efficiency.

3. DYNAMIC MODULATION SCALING (DMS)

When considering energy-aware techniques beyond shutdown, such as DMS, we require some more insight into the operation of radios. In this section, we introduce the basics of modulation scaling.

3.1 Energy and Delay Breakdown

The first step in understanding DMS is investigating how energy and throughput depend on the modulation level in a digital wireless communication system. In order to transmit information, bits are coded into channel symbols, which correspond to different waveforms [Proakis 1995]. The number of possible waveforms determines how many bits are coded into one symbol, which is given by the modulation level b, expressed in number of bits per symbol. The average time to transmit 1 bit over the channel is the inverse of the average bit rate R_b, and is given by (1), where R_S is the symbol rate (number of symbols that are transmitted per second).

$$T_{\text{bit}} = \frac{1}{R_b} = \frac{1}{b \cdot R_s}.$$ (1)

The energy associated with the transmission of 1 bit can be expressed as (2). The power needed to generate the information carrying electro-magnetic waves is delivered mainly by the power amplifier, and is denoted by the *transmit power* P_S. The remainder of the power consumption of the radio is lumped into P_E, the *electronics power*.

$$E_{\text{bit}} = [P_S + P_E] \cdot T_{\text{bit}}.$$ (2)

We note that (2) represents the energy consumption of the transmitter device, but it can also be used for the receiver, by setting P_S equal to zero. The value of P_E is not necessarily the same as that of the transmitter, of course.

Table I. Scaling Function $f(b)$ and Γ for Different Modulation Schemes

	2^b-QAM	2^b-PSK	2^b-PAM
$f(b)$	$2^b - 1$	$\sin\left(\frac{\pi}{2^b}\right)^{-2}$	$\frac{2^{2b}-1}{3}$
Γ	$1/3 \cdot \left[Q^{-1}\left(\left(1 - \frac{1}{2^{b/2}}\right)^{-1} \cdot \frac{b \cdot \text{BER}}{4} \right) \right]^2$	$1/2 \cdot \left[Q^{-1}\left(\frac{b \cdot \text{BER}}{2} \right) \right]^2$	$1/2 \cdot \left[Q^{-1}\left(\left(1 - \frac{1}{2^b}\right)^{-1} \cdot \frac{b \cdot \text{BER}}{2} \right) \right]$

Expressions (1) and (2) give the delay and energy associated with transmitting a bit over the wireless link, and therefore operate on the level of the OSI physical layer [Tenenbaum 1990]. These bits do not simply represent the bare application information (so-called "useful bits"), but contain contributions from higher-layer overhead, such as headers, channel coding, training sequences, control packets, and so on [Lettieri et al. 1999]. As we focus here on DMS, we keep all other parameters and protocol settings unchanged. The energy and delay per "useful bit" are thus a linear function of their corresponding values on the physical layer, such that we can focus on (1) and (2). We note that energy-efficient techniques on the other layers are still perfectly applicable and beneficial in addition to DMS.

To analyze DMS, we need to derive the detailed relationship between the energy and the modulation level. The scheme that is probably most amendable to scaling, due to its ease of implementation and analysis, is quadrature amplitude modulation (QAM) [Balachandran et al. 1999; Ue et al. 1998; Webb and Steele 1995]. The resulting bit error rate (BER) is well approximated by (3) [Proakis 1995]. In this equation, E_N is the noise energy per symbol, factor A contains all transmission loss components, and η is the linearized efficiency of the power amplifier. The $Q(\cdot)$ function is defined as (4).

$$\text{BER} = \frac{4}{b} \cdot \left(1 - \frac{1}{2^{b/2}}\right) \cdot Q\left(\sqrt{3 \cdot \frac{A \cdot \eta}{2^b - 1} \cdot \frac{P_S}{E_N \cdot R_S}} \right) \qquad (3)$$

$$Q(x) = \frac{1}{\sqrt{2\pi}} \cdot \int_x^\infty \exp\left(-\frac{1}{2} \cdot t^2\right) dt = \frac{1}{2} \cdot \text{erfc}\left(\frac{x}{\sqrt{2}}\right). \qquad (4)$$

By solving for the transmit power, we obtain (5), where parameter C_S is defined as (6). The expressions for $f(b)$ and Γ are listed in the first column of Table I.

$$P_S = C_S \cdot f(b) \cdot R_S \qquad (5)$$

$$C_S = \frac{E_N}{A \cdot \eta} \cdot \Gamma. \qquad (6)$$

E_N is a function of the receiver implementation and the operating temperature. A depends on distance and the propagation environment, and can vary with time. Neither of them varies with b. In addition, due to the $Q^{-1}(.)$ function, Γ is only very weakly dependent on b. Although we do take this dependency into account in the simulations, C_S is thus approximately a constant when we scale the modulation (i.e., vary b). The main benefits from modulation scaling are due to $f(b)$.

ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003.

169
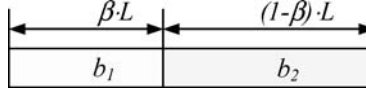
Fig. 1. Fractional modulation-level settings.

The power consumption of the electronic circuitry, which is largely analog, P_E, can be written as (7) [Cho and Samueli 2000; Schurgers et al. 2001a]. C_E is a constant that depends on the radio architecture, the circuit implementation, and the semiconductor technology.

$$P_E = C_E \cdot R_S. \tag{7}$$

With (5) and (7), expression (2) becomes an explicit function of the modulation level:

$$E_{\text{bit}} = [C_S \cdot f(b) + C_E] \cdot \frac{1}{b}. \tag{8}$$

The ratio of parameters C_S and C_E can thus be viewed as an indication of the relative importance of the transmit power versus the electronics power. In a similar way, we can derive expressions for phase shift keying (PSK) and pulse amplitude modulation (PAM) [Proakis 1995]. With the appropriate definitions of $f(b)$ and Γ as in Table I, (8) remains valid. In general, DMS is applicable to other scalable modulation schemes as well.

3.2 Granularity Effects

In the previous section, we implicitly assumed the modulation level could be varied continuously. However, strictly speaking, the expressions in Table I are only valid for integer values of b. In the case of QAM, they are exact only for even integers, but are reasonable approximations when b is odd (so-called "nonsquare" constellations) [Proakis 1995].

Furthermore, it is impractical to change the modulation level at arbitrary time instants, since both sender and receiver need to know the exact modulation scheme that is used. This requires a kind of negotiation between the two, resulting in protocol overhead. It makes sense to limit the rate of adaptation, for example, by restricting it to periodic instants or the start of packet transmissions. The implication toward implementation is that the receiver has to be designed such that it can appropriately reconfigure its processing for the right modulation level. This is one crucial difference between modulation scaling and voltage scaling, which, as we will see later, also affects the power management strategies.

However, in between two modulation updates, we can define a fractional modulation level. For example, the first half of the packet could be sent with modulation b_1 and the second half with b_2, see Figure 1. As a result, the average energy and delay per bit are a linear interpolation between the corresponding values of the two modulation levels, as is apparent from (9) and (10).

$$E_{\text{bit}} = \frac{E_{\text{packet}}}{L} = E_{\text{bit}}^1 \cdot \beta + E_{\text{bit}}^2 \cdot (1 - \beta) \tag{9}$$
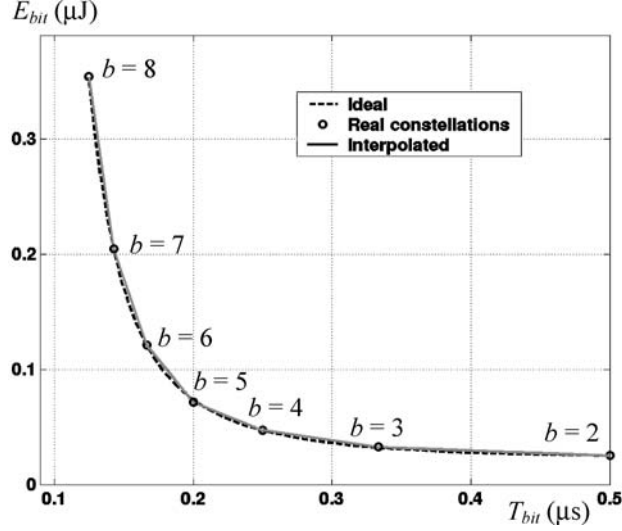
Fig. 2.   Energy-delay trade-off for QAM.

Table II. Simulation Settings

R_S	1 MH$_Z$
BER	10^{-5}
C_S (4-QAM)	12 nJ
C_E	15 nJ

$$T_{\text{bit}} = \frac{T_{\text{packet}}}{L} = T_{\text{bit}}^1 \cdot \beta + T_{\text{bit}}^2 \cdot (1 - \beta). \qquad (10)$$

Each modulation scheme has a $b_{\min}$, which is the minimum practically achievable modulation. If we are allowed even more delay per bit, we can shutdown the radio for a while ($b = 0$). However, as mentioned before, shutdown does not reduce the energy per bit. The maximum modulation $b_{\max}$ is only bounded by implementation choices and maximum transmit power. Between $b_{\min}$ and $b_{\max}$, the modulation can be scaled with granularity δ, which is a design choice. This minimum and maximum level and granularity effect are similar to what is encountered in voltage scaling as well.

Figure 2 illustrates the energy-delay trade-off for QAM, with the numerical values from Table II. C_S varies slightly with the modulation level, and the value at operating point $b = 2$ is listed in the table. The curve labeled "ideal" corresponds to (1) and (8). The circles indicate constellations that can be realized, and the solid line gives the values that are obtained through the interpolation we just explained. For QAM, $b_{\min}$ is equal to 2. We see that we can use (1) and (8) for analysis purposes as very tight approximations to what is practically realizable.
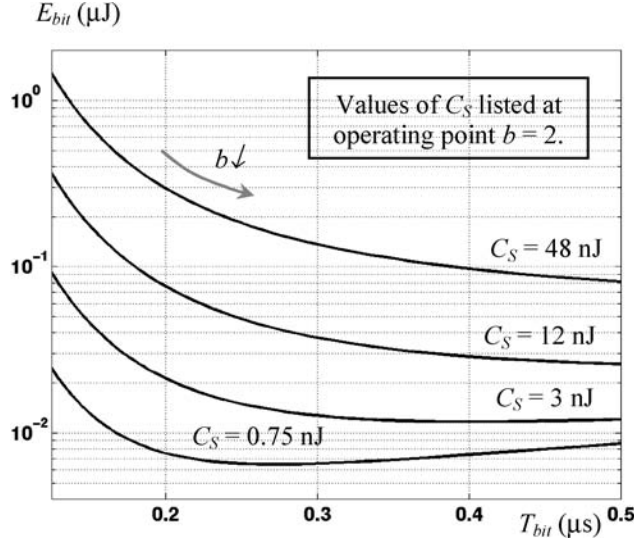
Fig. 3. Energy-delay trade-off for different C_S.

3.3 Region of Modulation Scaling

Figure 3 shows the same trade-off for QAM for different values of C_S. The values of C_S shown in the figure are for operating point $b = 2$, that is, 4-QAM, and correspond to P_S varying from 2.25 mW to 144 mW. The other settings are those of Table II. As C_E is kept constant, a higher value of C_S thus indicates that the transmit power portion becomes more dominant compared to the electronics power portion.

DMS essentially utilizes the effect that, by varying the modulation level, energy can be traded off versus delay. At the left side of the figure, lowering b reduces the energy, at the cost of an increased delay. Alternatively, the power versus speed is convex in this region. Scaling beyond the point of minimum energy clearly does not make sense, as both energy and delay would increase. The operating region of DMS therefore corresponds to the portion of the curves to the left of their minimum energy point. It is clear that in this region, scaling is superior to shutdown, as the metric E_{bit} is the total energy per bit, which will not change if the transmission is followed by a period of shutdown. We can also verify that the energy–delay curves are convex, such that a uniform stretching of the transmissions is the most energy-efficient (due to Jensen's inequality).

From Figure 3, we see that DMS is more useful for situations where C_S is large, or in other words, where the transmit power dominates the electronics power. This is true except for communication systems with a very short transmission range. Also note that the electronics power behaves similarly to the leakage current in digital circuits. As leakage becomes more dominant with shrinking CMOS device dimensions [Sinha and Chandrakasan 2000], DVS will be faced with similar issues of operating region, as observed here.

ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003.

172

Power Managemer

Table III. Modulation Bounds

$b_{\min}$ (bits/symbol)	$b_{\max}$ (bits/symbol)	δ (bits/symbol)
2	8	0.5

3.4 Difference Between DMS and DVS

Despite the analogies between DVS and DMS, there are two important differences. First, a change in modulation level requires communication between sender and receiver. As explained in Section 3.2, the sender cannot decide on a new modulation setting midway through the packet transmission. Packet scheduling therefore exhibits this inherent time-granularity effect and can only be nonpreemptive. However, the exact packet size is known at the start of the transmission, in contrast to the task execution time in processors [Raghunathan et al. 2001]. We detail these points in Section 4.1.

Second, the wireless channel may vary over time, which means that the factor A in (6) fluctuates. These variations have to be taken into account in the packet-scheduling engine. This is akin to a processor where capacity (in MIPS) would change over time, for example, due to interrupt handling. This phenomenon is indeed complex, but is beyond the designers' control in radio power management.

4. ENERGY-AWARE PACKET SCHEDULING

Just like DVS has driven energy-aware task scheduling beyond shutdown-based approaches [Burd et al. 2000; Govil et al. 1995; Gruian 2001; Gutnik and Chandrakasan 1997; Krishna and Lee 2000; Manzak and Chakrabarty 2000; Nielsen et al. 1994; Shin and Choi 1999; Weiser et al. 1994; Yao et al. 1995], DMS paves the way toward new energy-aware packet-scheduling policies. The body of literature dealing with packet scheduling is vast, and, in principle, is suitable to be extended toward energy-aware versions using DMS [Demers et al. 1989; Lu et al. 1997; Parekh and Gallager 1994]. However, many challenges lie ahead, since such radio power management must deal with both traffic load and channel variations. In the next subsections, we describe two basic scheduling approaches to illustrate some of the challenges. They each highlight one of two different issues, namely the presence of deadlines and channel variations.

In all simulations reported in this section, a special field in the packet header, encoded with 4-QAM, is used to communicate the modulation level for the rest of the packet to the receiver. We chose this option for its simplicity and fault tolerance, assuming we have control over the protocol stack. If this is not available, an alternative is to use separate control packets. The DMS scheduler, residing at the link-layer, might still require channel quality information from the lower layers, similar to other adaptive protocols [Balachandran et al. 1999; Lettieri et al. 1999; Thoen et al. 2000]. Table III defines the possible values, which can be encoded into 4 bits. This overhead for each packet is incorporated in all simulation results.

4.1 Real-Time Scheduling in a Time-Invariant Channel

In this section, we consider the problem of scheduling multiple real-time traffic streams, while the wireless channel does not vary in time. In each stream, packets arrive periodically and have a deadline by which they have to be sent. This is a valid model for multimedia streams. We choose the deadline of each packet to be the arrival time of the next packet in that stream. The size of the individual packets might vary (e.g., in MPEG video or perceptual audio codecs), but there is a maximum known packet size for each stream. As mentioned before, the packet scheduler should not interrupt an ongoing transmission such that the policy has to be nonpreemptive. This is a new constraint compared to the energy-aware task scheduling policies for RTOS, which are all preemptive (i.e., tasks can be suspended and resumed later on) [Burd et al. 2000; Gruian 2001; Krishna and Lee 2000; Manzak and Chakrabarty 2000; Shin and Choi 1999; Yao et al. 1995]. The setup we consider here is analogous to the DVS work described in Raghunathan et al. [2001]. Besides the preemptive nature, the scheduling algorithm of Raghunathan et al. [2001] deals with task-length variability in a stochastic fashion, as the exact execution time is not known at the start of the task. However, in packet scheduling, the exact packet length is known at the start of the transmission.

A condition that guarantees schedulability for nonpreemptive scheduling is derived in Jeffay et al. [1991]. When it is satisfied, earliest-deadline-first scheduling (EDF) always results in a valid schedule. An optimal energy-aware scheduling routine is too computationally intensive as the problem is NP-complete; but we propose a practical algorithm, which consists of two steps [Schurgers et al. 2001b].

1. *Admission step*: When a new stream is admitted to the system, we calculate a static scaling factor, α_{static}, assuming all packets are of maximum size. This factor is the minimum possible such that if the modulation setting for each packet were scaled by it, the schedulability test would still be satisfied. In other words, it computes the slowest transmission speed at which all the packet streams are just schedulable.

2. *Adjustment step*: At run-time, packets are scheduled using EDF. Before transmission starts, the actual size of each packet is known. We calculate an additional scaling factor, α_{dyn}, such that the transmission finishes when that of a maximum size packet would have. Since step 1 assumes the maximum packet size, the schedulability is guaranteed. If the system were still idle after the packet transmission, we would stretch the transmission until the packet's deadline or the arrival time of a new packet (which is known due to the periodic nature of the traffic). This extra scaling factor is called α_{stretch}.

The scheduler combines all three scaling factors to get the overall modulation that is used for the current packet [Schurgers et al. 2001b]. To evaluate the performance of our scheme, we have carried out a number of simulations. The basic settings and modulation settings are those of Tables II and III, respectively. The

ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003.

174

Table IV. Number of Streams with Given Period

	20 ms	25 ms	40 ms	50 ms
$U = 0.82$	8	6	4	4
$U = 0.77$	8	6	2	4
$U = 0.74$	8	4	4	4
$U = 0.69$	6	4	6	4
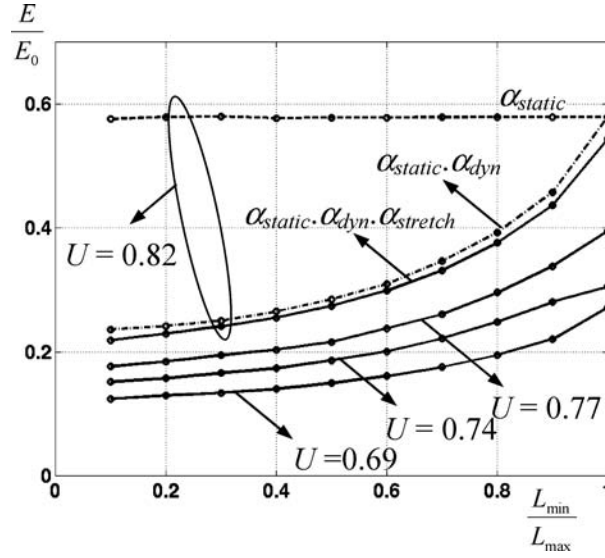
Fig. 4. Relative energy for real-time energy-aware packet scheduling.

size of each packet is independently chosen in a uniformly random fashion between the maximum packet size L_{max} of 8000 bits and a minimum value L_{min}, which is a parameter we vary in our simulations. We consider four different scheduling scenarios. For each of them, a row in Table IV lists the number of streams with a given period and the resulting total link utilization when all packets are of maximum size. For example, in the fourth scenario in Table IV, there are six streams with a period of 20 ms, four with a period of 25 ms, six with a period of 40 ms, and four with a period of 50 ms, resulting in a link utilization U of 0.69 when $L_{min} = L_{max}$.

Figure 4 plots the energy consumption of our scheduling scheme, normalized against one without scaling ($b = b_{max}$ at all times). For $U = 0.82$, we have separated the effect of the different scaling factors. When only using α_{static}, the transmissions are slowed down uniformly without exploiting the run-time packet length variations. These are leveraged through α_{dyn}, where the energy decreases as the packet size variation increases (L_{min} decreases). The effect of α_{strech} is marginal here. For the other utilizations, we only show the results when combining all three scaling factors. As expected, more energy savings are achieved when the utilization is lower.

175

The power management scheme described here, essentially exploits traffic load variations on two levels to introduce energy awareness.

1. Variations in overall utilization are handled by the admission step in our algorithm through α_{static}. These are due to changes in number of streams, which are likely to occur over relatively large time scales.
2. Variations in individual packet sizes on the other hand occur at much smaller time scales. These cannot be handled during admission, but are exploited in the adjustment step of our algorithm.

4.2 Nonreal-Time Scheduling in Time-Variant Channel

In this section, we highlight a different issue in radio power management, namely the effects of time variations in the wireless channel. The closest equivalent in CPU task scheduling is a time-varying processor capacity. To introduce some of the challenges, we consider a rudimentary scenario: the transmission of a single data stream that has no hard deadline associated with it, but only an average data rate constraint. This model is useful in the case of a file transfer, for example. In Schurgers and Srivastava [2002], we also illustrate how this specification can be used to provide a soft real-time constraint.

As discussed in Section 3.1, the transmission loss A captures the effect of the wireless channel. In the presence of time variations, this factor is split up into two components as in (11), where $\overline{A}$ represents the long-term average value and α contains the normalized time variations. The behavior of the gain factor α can be characterized by two statistics: a probability density function and a Doppler rate, which describes the time correlation [Jakes 1994; Proakis 1995].

$$A = \overline{A} \cdot \alpha \tag{11}$$

In all techniques that adapt a radio parameter to channel variations, an estimate of the current channel condition is needed [Balachandran et al. 1999; Lettieri et al. 1999]. This is obtained through channel estimation, which is updated regularly. The update rate f_{update} is chosen such that the channel remains approximately constant between updates, yet the overhead of the estimation process is limited. In addition, predictive compensation techniques can be applied [Thoen et al. 2000]. Although deciding the appropriate update rate, as a compromise between overhead and performance degradation, is an important issue, a detailed study falls outside the scope of this paper.

In the previous section, DMS turned into a scheduling problem because of the interaction between multiple streams. Here, we only have one stream, but the presence of a time-varying channel again makes the choice of the best value of b a scheduling issue. The current decision critically depends on how good or bad the channel will be, that is, whether it is more energy efficient to send now or later. However, in the specific scenario we consider here, namely that the location of the average throughput is the only additional constraint, the problem can be greatly simplified. In Schurgers and Srivastava [2002], we prove that there exists a set of thresholds d_i that directly link the current channel condition to the optimal choice of b. Equation (12) presents a direct generalization of the results presented in Schurgers and Srivastava [2002]. It is not valid for very

high values of C_E/C_S, but holds the ones chosen here [Schurgers 2002]. We refer to the resulting energy-efficient DMS packet-scheduling policy as "loading in time," as it was inspired by adaptive bit-loading techniques for a class of radio systems called multicarrier systems [Chow et al. 1995; Hughes-Hartogs 1987]. Note that only those values of b are assigned that did not result from the interpolation of Section 3.2. The reason why they are not included in the optimal assignment is that the energy versus delay curve is no longer strictly convex there [Schurgers 2002].

$$\begin{cases} 0 \le \alpha < d_1 \Rightarrow b = 0 \\ d_i \le \alpha < d_{i+1} \Rightarrow b = b_{\min} + (i-1), \quad i = 1..(K-1) \\ d_K \le \alpha < \infty \Rightarrow b = b_{\max} \end{cases}$$

$$K = 1 + b_{\max} - b_{\min}. \tag{12}$$

Furthermore, the thresholds d_i are mutually related according to (13) and (14) [Schurgers 2002]. The fact that both transmit and electronics power are zero when $b = 0$ is taken into account here.

$$d_i = \max[\theta \cdot 2^{i-1}, d_1] \qquad i = 2..K \tag{13}$$

$$\theta = b_{\min} \cdot \left[\frac{2^{b_{\min}} - 1}{d_1} + \frac{C_E}{C_S} \right]^{-1} \cdot 2^{b_{\min}-1}. \tag{14}$$

There is only one independent parameter left, which can be solved from the constraint on the desired average data rate b_{av}, expressed in average number of bits per symbol. This constraint can be written as (15), where $G(x)$ is defined in (16) [Schurgers and Srivastava 2002]. Here, $F(x)$ is the cumulative distribution function of the gain factors α.

$$b_{\mathrm{av}} = b_i \cdot G(d_1) + \sum_{i=2}^{K} G(d_i) \tag{15}$$

$$G(x) = 1 - F(x). \tag{16}$$

The thresholds thus only depend on the statistics of the wireless channel, which can be estimated online. We no longer have to know the exact behavior of the channel over time to achieve the energy-optimal scheduling policy. Figure 5 shows the simulated performance of this radio power management scheme versus the average throughput constraint. As before, we used the values of Tables II and III. The channel exhibits the correlated Rayleigh fading, such that $G(x)$ is given by (17). This corresponds to the case where there is no line-of-sight path between sender and receiver, which is the predominant model used in literature [Proakis 1995]. The time correlation of the channel is characterized by a Doppler rate of 50 Hz, and simulated as proposed in Jakes [1994].

$$G(x) = \exp(-x) \tag{17}$$

We selected an update rate f_{update} of 1 kHz for channel estimation, the overhead of which is included in the reported simulation results. The maximum

ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003.

177

E_{bit} (μJ)
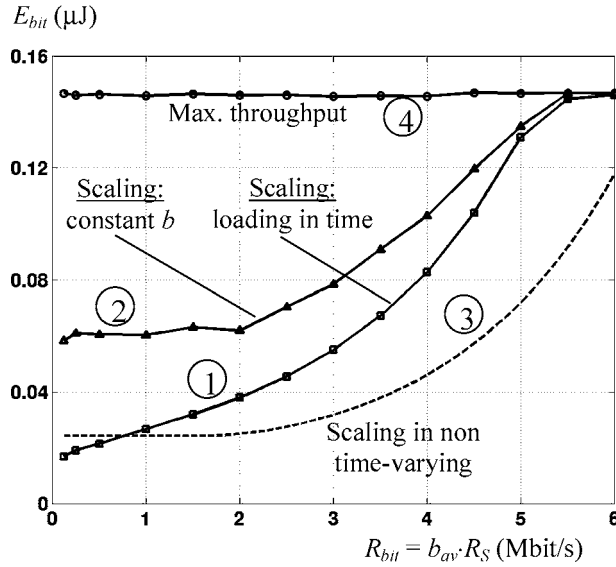
Fig. 5.   Energy versus average bit rate for time-varying channel (Rayleigh fading).

possible transmit power is 1 W. Curve 1 in Figure 5 plots the behavior of the "loading in time" scheduling policy that we described in this section. It is superior to scaling with "constant b" (curve 2), where the modulation is uniformly slowed down based on the average throughput, but channel variations are not taken into account. The difference between curves 2 and 3, which shows the same uniform scaling in a nontime-varying channel, illustrates the performance degradation associated with channel variations. Beyond $b_{\min} = 2$ bits/symbol, we resort to shutdown, and both these curves flatten out, which is as expected from our discussion in Section 3.2. However, curve 1 keeps on decreasing when lowering b_{av}, and can even outperform scaling in a non-time-varying channel (curve 3). The reason is that we still use shutdown, but only the very best time intervals (with $\alpha > 1$) are selected to send information. For curve 2, the shutdown was periodic, without taking the channel into account.

Finally, we also compare the performance to a scheme that is not energy-aware, but tries to achieve a "maximum throughput" possible. In this case, b is adapted to yield its maximum value without violating the maximum transmit power [Balachandran et al. 1999; Ue et al. 1998; Webb and Steele 1995]. As this is only based on the current channel condition, scheduling issues never arise. The benefits of energy-awareness, where a reduced throughput requirement is leveraged to yield energy savings, are again substantial.

5. CONCLUSIONS

Energy efficiency is gaining importance as a system design consideration, especially in portable communication devices. Radio-level power management based on shutdown simply turns off the radio when it is not used. However,

178

scaling-based techniques have the potential to be vastly superior in terms of energy savings, but necessitate more intricate power management. Packet scheduling has to be rethought to include energy-awareness, similar to the way voltage scaling prompted the search for energy-aware task scheduling. Numerous wireless packet scheduling techniques exist, which are all candidates to make it into an energy-aware version.

In this paper, we focused on dynamic modulation scaling, DMS, as just one of the possible radio control knobs that can be used for scaling. Even in the elementary scenarios we considered here, it illustrates the intricacies and challenges of energy-aware packet scheduling. These will be compounded when stringent delay/throughput constraints have to be satisfied in the presence of a time-varying wireless channel, which is inherently stochastic in nature. It is expected that resulting policies will not be able to guarantee service, but be characterized by reliability bounds, where the tightness of the bounds can be traded off for energy savings. Radio power management therefore has to take both traffic and channel aspects into account simultaneously. Other research challenges that remain are enhancing these strategies to also handle multiple devices on a shared channel. This likely requires a distributed manipulation of the radio control knobs, by incorporating it in the medium access control layer.

The resulting radio power management eventually has to be integrated into a system-wide solution, where energy and latency can be traded off across the computation–communication subsystem boundaries. The overall vision is thus a coordinated power management, intermixing both task and packet scheduling policies in a networked system. Furthermore, these ideas, while presented here in the context of radios, are potentially generalizable to wired communications as well, which may offer other control knobs for power-speed scaling.

REFERENCES

BALACHANDRAN, K., KADABA, S. R., AND NANDA, S. 1999. Channel quality estimation and rate adaptation for cellular mobile radio. *IEEE Journal on Selected Areas in Communications 17*, 7, 1244–1256.

BENINI, L. AND DE MICHELI, G. 1997. *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer, Norwell, MA.

BENINI, L., BOGLIOLO, A., AND DE MICHELI, G. 1999. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on VLSI Systems 8*, 3, 813–833.

BURD, T., PERING, A., STRATAKOS, A., AND BRODERSEN, R. 2000. A dynamic voltage scaled microprocessor system. *IEEE Journal of Solid-State Circuits 35*, 11, 1571–1580.

CHANDRAKASAN, A., SHENG, S., AND BRODERSEN, R. 1992. Low-power CMOS digital Design. *IEEE Journal of Solid-State Circuits 27*, 4, 473–484.

CHO, K. AND SAMUELI, H. 2000. A 8.75-MBaud single-chip digital QAM modulator with frequency-agility and beamforming diversity. In *Proceedings CICC'00*. Orlando, FL (May), 27–30.

ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003.

179

CHOW, P., CIOFFI, J., AND BINGHAM, J. 1995. A practical discrete multitone transceiver loading algorithm for data transmission over spectrally shaped channels. *IEEE Trans. on Communications 43*, 2, 773–775.

DEMERS, A., KESHAV, S., AND SHENKER, S. 1989. Analysis and simulation of a fair queueing algorithm. *Computer Communication Review 19*, 4, 1–12.

GOVIL, K., CHAN, E., AND WASSERMAN, H. 1995. Comparing algorithms for dynamic speed-Setting of a low-power CPU. In *Proceedings MobiCom'95*. Berkeley, CA (Nov.), 13–25.

GRUIAN, F. 2001. Hard real-time scheduling for low energy using stochastic data and DVS processor. In *Proceedings ISLPED'01*. Huntington Beach, CA (Aug.), 46–51.

GUTNIK, V. AND CHANDRAKASAN, A. 1997. Embedded power supply for low-power DSP. *IEEE Transactions on VLSI Systems 5*, 4, 425–435.

HUGHES-HARTOGS, D. 1987. Ensemble modem structure for imperfect transmission media. U.S. Patents, nos. 4,679,227 (July 1987), 4,731,816 (March 1988), 4,833,796 (May 1989).

JAKES, W. 1994. *Microwave Mobile Communication*. John Wiley, New York.

JEFFAY, K., STANAT, D., AND MARTEL, C. 1991. On non-preemptive scheduling of periodic and sporadic tasks. In *Proceedings IEEE RTSS'91*. San Antonio, TX (Dec.), 129–139.

KRISHNA, C. AND LEE, Y. 2000. Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems. In *Proceedings RTAS'00*. Washington, DC (June), 156–165.

LETTIERI, P., SCHURGERS, C., AND SRIVASTAVA, M. 1999. Adaptive link layer strategies for energy efficient wireless networking. *Wireless Networks 5*, 5, 339–355.

LORCH, J. AND SMITH, A. 1998. Software strategies for portable computer energy management. *IEEE Personal Communications 5*, 3, 60–73.

LU, S., BHARGHAVAN, V., AND SRIKANT, R. 1997. Fair scheduling in wireless packet networks. *Computer Communication Review 27*, 4, 63–74.

MANZAK, A. AND CHAKRABARTY, C. 2000. Variable voltage task scheduling for minimizing energy or minimizing power. In *Proceedings ICASSP'00*. Istanbul, Turkey (June), 3239–3242.

NIELSEN, L., NIESSEN, C., SPARSØ, J., AND VAN BERKEL, K. 1994. Low power operation using self-timed circuits and adaptive scaling of the supply voltage. *IEEE Transactions on VLSI Systems 2*, 4, 391–397.

PAREKH, A. AND GALLAGER, R. 1994. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking 2*, 2, 137–150.

PEDRAM, M. 2001. Power optimization and management in embedded systems. In *Proceedings ASP-DAC 2001*. Yokohama, Japan, 239–244.

PRABHAKAR, B., BIYIKOGLU, E., AND EL GAMAL, A. 2001. Energy-efficient transmission over a wireless link via lazy packet scheduling. In *Proceedings Infocom'01*. Anchorage, AK (April), 386–394.

PROAKIS, J. 1995. *Digital Communications*. McGraw-Hill Series in Electrical and Computer Engineering. McGraw-Hill New York.

RAGHUNATHAN, V., SCHURGERS, C., PARK, S., AND SRIVASTAVA, M. 2002. Energy-aware wireless sensor networks. *IEEE Signal Processing Magazine 19*, 2, 40–50.

RAGHUNATHAN, V., SPANOS, P., AND SRIVASTAVA, M. 2001. Adaptive power-fidelity in energy aware wireless systems. In *Proceedings RTSS'01*. London, UK (Dec.), 106–115.

SCHURGERS, C. 2002. Energy-aware wireless communications. Ph.D. dissertation, University of California at Los Angeles.

SCHURGERS, C., ABERTHORNE, O., AND SRIVASTAVA, M. 2001a. Modulation scaling for energy aware communication systems. In *Proceedings ISLPED'01*. Huntington Beach, CA (Aug.), 96–99.

SCHURGERS, C., RAGHUNATHAN, V., AND SRIVASTAVA, M. 2001b. Modulation scaling for real-time energy aware packet scheduling. In *Proceedings Globecom'01*. San Antonio, TX (Nov.), 3653–3657.

SCHURGERS, C. AND SRIVASTAVA, M. 2002. Energy efficient wireless scheduling: Adaptive loading in time. In *Proceedings WCNC'02*. Orlando, FL (Mar.), 706–711.

SHIN, Y. AND CHOI, K. 1999. Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings DAC'99*. New Orleans, LA (June), 134–139.

SINHA, A. AND CHANDRAKASAN, A. 2000. Energy aware software. In *Proceedings VLSI Design 2000*. Calcutta, India (Jan.), 50–55.

ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003.

180

SRIVASTAVA, M., CHANDRAKASAN, A., AND BRODERSEN, R. 1996. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Transactions. on VLSI Systems 4*, 1, 42–55.

TENENBAUM, A. 1990. *Computer Networks*. Prentice-Hall, Englewood Cliffs, NJ.

THOEN, S., VAN DER PERRE, L., GYSELINCKX, B., ENGELS, M., AND DE MAN, H. 2000. Predictive adaptive loading for HIPERLAN II. In *Proceedings VTC'00 Fall*. Boston, MA (Sept.), 2166–2172.

UE, T., SAMPEI, S., MORINAGA, N., AND HAMAGUCHI, K. 1998. Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD System for High-Bit Rate Wireless Data Transmission. *IEEE Transactions on Vehicular Technology 47*, 4, 1134–1147.

WANG, H. AND MANDAYAM, N. 2001. Delay and energy constrained dynamic power control. In *Proceedings Globecom'01*. San Antonio, TX (Nov.), 1287–1291.

WEBB, W. AND STEELE, R. 1995. Variable rate QAM for mobile radio. *IEEE Transactions on Communications 43*, 7, 2223–2230.

WEISER, M., WELCH, B., DEMERS, A., AND SHENKER, B. 1994. Scheduling for reduced CPU energy. In *Proceedings USENIX Symposium on Operating Systems Design and Implementation*. Monterey, CA (Nov.), 13–23.

YAO, F., DEMERS, A., AND SHENKER, S. 1995. A scheduling model for reduced CPU energy. In *Proceedings 36th Annual Symposium on Foundations of Computer Science*. Milwaukee, WI (Oct.), 374–385.

ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003.

181

Energy Aware Wireless Systems with Adaptive Power-Fidelity Tradeoffs*

Vijay Raghunathan, Cristiano L. Pereira†, Mani B. Srivastava, and Rajesh K. Gupta‡
Department of EE, UC Los Angeles, CA 90095
†Department of ICS, UC Irvine, CA 92697
‡Department of CSE, UC San Diego, CA 92093
{vijay, mbs}@ee.ucla.edu   cpereira@ics.uci.edu   gupta@cs.ucsd.edu

Abstract

Wireless embedded systems, such as multimedia terminals, sensor nodes, *etc.*, combine soft real-time constraints on computation and communication with requirements of long battery lifetime. Energy aware system operation, and not just low power hardware design, is an crucial requirement for these systems. This paper presents an operating system directed dynamic voltage scaling (DVS) technique that enhances the energy awareness of these systems. A key feature of our technique, not addressed in prior work, is that it yields an adaptive tradeoff between energy consumption and system delity .

The proposed algorithm exploits two observations about the operating scenario of wireless embedded systems, namely, (i) they are designed to operate resiliently in the presence of varying delity (data losses, delays, and errors over the wireless link), and (ii) they exhibit signi cant correlated variations in their computation and communication processing load due to underlying time-varying physical phenomena. These observations enable the proposed algorithm to proactively manage energy resources by predicting processing requirements, thereby trading off energy consumption against system delity . The supply voltage and processor clock frequency are set according to the predicted execution time of a given task instance, and an adaptive feedback mechanism keeps the system delity (deadline misses) within speci cations.

We present the theoretical framework underlying our approach in the context of a static priority based preemptive task scheduler. We show through extensive simulations that our technique results in signi cant energy savings compared to state-of-the-art DVS schemes with minimal loss in system delity . We have implemented our technique into the *eCos* real-time operating system running on an Intel XScale based variable voltage platform. Experimental results obtained through actual power measurements on this platform con rm the effectiveness of our technique.

1 INTRODUCTION

Embedded systems are increasingly becoming networked, often wirelessly, and demand an integration of high-performance computing and wireless communication capabilities, under real-time constraints. Examples of such systems include mobile multimedia terminals with wireless LAN cards, 3G cellular phones, ad-hoc networks of wireless sensors, wireless toys and robots, *etc*. The battery operated nature of these systems requires them to be designed and operated in a highly energy efficient manner to maximize their lifetime.

The drive to prolong system lifetime has resulted in the development of several low power hardware design techniques [1, 2]. In addition to using low power hardware components, managing the various system resources in an energy aware manner can further reduce energy consumption considerably, thereby extending battery life. One commonly used approach for energy aware system operation is Dynamic Power Management (DPM) [3], in which unused system components are shutdown or sent into low power states. The goal of system-level DPM techniques is to obtain an optimized power-state transition policy [4, 5, 6, 7]. An alternative, and more effective when applicable, technique used to increase energy efficiency is Dynamic Voltage Scaling (DVS) [8, 9, 10, 11, 12, 13, 14], in which the supply voltage and clock frequency of the processor are changed dynamically to just meet the instantaneous performance requirement.

In order to adapt to constantly evolving standards, a significant fraction of the functionality of wireless embedded systems is implemented as software running under a real time operating system (RTOS). Each application (*e.g.*, video encoding/decoding, data encryption/decryption, layers of the protocol stack, *etc.*) runs as a concurrent task under the RTOS. The RTOS is uniquely poised to efficiently implement DPM and DVS policies due to the following reasons: (i) since the RTOS coordinates the execution of the various application tasks, it has global information about the performance requirements and timing constraints of all the applications, and (ii) the RTOS can directly control the underlying hardware platform, fine-tuning it to meet specific system requirements.

In this paper, we adopt an RTOS directed approach to power management and DVS, and enable energy aware system operation by exploiting the following characteristics of wireless embedded systems:

- **Most wireless systems are resilient to packet losses and errors.** The operating scenarios of these systems invariably have losses and errors associated with them (*e.g.*, noisy wireless channel conditions lead to packet losses and errors), and the application layer is designed to be tolerant to these impairments. Most wireless systems, therefore, are "soft" real-time systems where a few deadline misses only lead to a small degradation in the user-perceived system fidelity.

- **Wireless embedded systems offer a power-fidelity tradeoff that can be tuned to suit application needs**. For example, the precision of the computation (*e.g.*, quality of the audio or video decoding) can be traded off against the power consumed for the computation. As another example, wireless channel induced errors can be reduced by increasing the transmission power of the radio.

- **Wireless embedded systems have time varying computational loads.** Performance analysis studies have shown that for typical wireless applications (*e.g.*, multimedia encoding/decoding, encryption/decryption, *etc.*), the instance to instance task execution time varies significantly, and is often far lower than the worst case

183

Program	Description	BCET	WCET
DES	Data encryption	73,912	672,298
DJPEG	JPEG decompression (128x96, color)	12,703,432	122,838,368
FDCT	JPEG forward DCT	5,587	16,693

Table 1: Variation in execution times (in clock cycles) for a few multimedia benchmarks [15]

execution time (WCET). Table 1 gives the WCET and best case execution time (BCET) for a few benchmarks [15], and clearly illustrates the large workload variability. Note that, even though the execution times vary signi cantly , they depend on data values that are obtained from physical real-world signals (*e.g.*, audio or video streams), and are likely to have some temporal correlation between them. This enables us to predict task instance execution times with reasonable accuracy.

1.1 Paper contributions

The contributions of this paper are threefold:

1. We present an enhanced x ed priority schedulability analysis for variable voltage systems that incorporates the timing overheads incurred due to voltage/frequency changes and processor shutdown/wakeup. This analysis can be used to accurately analyze the schedulability of *non-concrete* periodic task sets, scheduled using the Rate Monotonic (RM) or the Deadline Monotonic (DM) priority assignment schemes.

2. We present a proactive DVS technique that exploits the above described characteristics of wireless embedded systems to achieve signi cant energy savings. A key feature of our technique, not addressed in prior work, is that it yields an adaptive tradeoff between energy consumption and system delity/quality .

3. Our DVS technique has been implemented into the kernel of the eCos [16] real-time operating system running on an Intel XScale [17] processor. We describe our implementation and present *measured* results to demonstrate the effectiveness of our technique. As part of our implementation, we have also developed a complete software architecture that facilitates application-RTOS interaction for ef cient DPM/DVS.

The proposed DVS technique exploits task runtime variation by predicting task instance execution times and performing DVS accordingly. Most previously proposed predictive DVS algorithms [18, 19] are processor utilization based, and hence perform poorly in the presence of latency (*i.e.*, deadline) constraints. In contrast, our technique is tightly coupled to the schedulability analysis of the underlying real-time scheduling scheme, thereby yielding signi cantly superior results than utilization based approaches. The few deadline misses that result from occasional mis-prediction are not catastrophic to system performance due to the inherent error-tolerant nature of these wireless systems. In addition, our algorithm uses a novel adaptive feedback mechanism to keep a tight check on the number of task deadline misses. This is done by monitoring recent deadline miss history, and accordingly adapting the prediction to be more aggressive/conservative. A thread based proactive DVS algorithm, which used the idea of execution time prediction, was presented in [23] for scheduling sporadic tasks with deadline constraints. Our work differs from the work presented in [23] in two ways, (i) while the algorithm of [23] permits missed deadlines, it does

not provide any control on the number of deadline misses. Through appropriate parameter selection, the proposed algorithm permits the user to control the number of deadline misses, and trade them off for increased energy savings, and (ii) the algorithm of [23] is a purely online one, and has a complexity of $O(n)$, where n is the number of tasks present in the system. In contrast, since the proposed algorithm is based on an ofine schedulability analysis, the complexity of the online component is $O(1)$, *i.e.*, the scheduler overhead is independent of the number of tasks in the system, resulting in better scalability. This can be signicant for systems with large number of tasks, especially since online DVS algorithms are executed during every context switch.

Finally, since our technique is based on the enhanced schedulability analysis mentioned above, it can even be used to schedule *non-concrete* task sets, where the initial phase offsets of tasks are unknown. This is unlike most existing variable voltage real-time scheduling schemes, which require the initial phase offsets of the tasks to be known in order to perform hyper-period scheduling.

1.2   Related work

Dynamically scaling the supply voltage was rst proposed for asynchronous circuits [8]. Since then, there has been extensive research in the area of DVS. This research has spurred (and, in turn, has been spurred by) the emergence of processors such as Intel's StrongARM [20] and XScale [17], and Transmeta's Crusoe [21] that were designed to support dynamic voltage and clock scaling.

Initial work on DVS was in the context of a workstation-like environment [9, 10] where average throughput is the metric of performance, and latency is not an issue since there are no real-time constraints. In these techniques, time is divided into 10-50 ms intervals, and the processor frequency and supply voltage are adjusted by the task-level scheduler based on the processor utilization over the preceding interval. Similar utilization based predictive techniques were studied in more detail in [18, 19]. A technique to derive an ofine minimum energy schedule, and an average rate heuristic for scheduling independent processes with deadlines was presented in [11]. Since then, numerous DVS techniques have been proposed for low power real-time task scheduling, both for uniprocessor [12, 13, 14, 18, 19, 22, 23, 24, 25, 26, 27, 28] as well as multiprocessor [29, 30, 31] systems. A detailed survey of these techniques is presented in [32]. Most uniprocessor DVS techniques [12, 13, 14, 24, 25, 26, 27, 28] perform voltage scheduling assuming that systems have hard deadline constraints, and several of them assume that each task instance executes for its WCET. Even the few schemes that allow deadline misses [18, 19] involve utilization based speed setting decisions, and hence exhibit poor real-time behavior. More importantly, they only consider systems with a single application executing (*i.e.*, a unitasking environment). The authors of [23] discuss a thread based proactive DVS technique for uniprocessor, multitasking systems. Similar to [23], the technique proposed in this work considers a uniprocessor, multitasking, real-time environment, and shows that permitting a few deadline misses leads to a signicant increase in energy savings and improves the energy scalability of the system.

The rest of this paper is organized as follows. Section 2 presents examples to illustrate the power management opportunities that exist during real-time task scheduling. Section 3 describes our system model. Section 4 presents the enhanced RM schedulability analysis for variable voltage systems. Section 5 discusses our adaptive power-delity tradeoff technique. Section 6 details our simulation based performance analysis. Section 7 describes our experimental setup and the implementation of our technique into eCos. Section 8 presents the conclusions.

2 BACKGROUND AND ILLUSTRATIVE EXAMPLES

Next, we describe the basic scheduling approach adopted in our work, and present examples to illustrate the DPM/DVS opportunities that arise during real-time task scheduling.

2.1 Task scheduling in RTOS

The task scheduler of an RTOS is responsible for scheduling a given set of tasks such that real-time constraints are satis ed. Schedulers differ in the type of scheduling policy they implement. A commonly used scheduling policy is Rate Monotonic (RM) scheduling [33]. This is a x ed-priority based preemptive scheduling scheme where tasks are assigned priorities in the inverse ratio of their time periods. Fixed priority based preemptive scheduling is commonly used in operating systems such as eCos, WinCE, VxWorks, QNX, uC/OS, *etc.*, that run on wireless embedded devices such as handheld multimedia nodes. An alternative scheduling scheme is Earliest Deadline First (EDF) scheduling [33], where task priorities are assigned dynamically such that task instances with closer deadlines are given higher priorities. EDF can schedule task sets with a higher processor utilization than can be scheduled by a static priority based scheduling scheme such as RM. However, dynamic priority based scheduling is also more complex to implement since task priorities keep changing during runtime. No matter which scheduling policy is used, the RTOS ensures that at any point of time, the currently active task is the one with the highest priority among the ready to run tasks. The problem of task scheduling on a uniprocessor system in the presence of deadlines is known to be solvable in polynomial time if the system is preemptive in nature [34]. However, the low energy scheduling problem was shown to be NP-Complete in [35], by a reduction from the "*Sequencing with deadlines and set-up times*" problem, described in [34]. Therefore, heuristic techniques have to be applied to minimize energy.

2.2 Power management opportunities during task scheduling

2.2.1 Static slack

It has been observed in many systems that, during runtime, even if all task instances run for their WCET, the processor utilization is often far lower than 100%, resulting in idle intervals. This slack that inherently exists in the system due to low processor utilization is henceforth referred to as **static slack**. It can be exploited to reduce energy consumption by statically slowing down the processor and operating at a lower voltage. While processor slowdown improves processor utilization, excessive slowdown may lead to deadline violations. Hence, the extent of slowdown is limited by the schedulability of the task set at the reduced speed, under the scheduling policy used. The following example illustrates the use of static slowdown and voltage scaling to reduce energy consumption.

Task	Time Period	WCET	Deadline
Audio decoding	60	10	60
Protocol processing	70	15	70
Video decoding	120	40	120

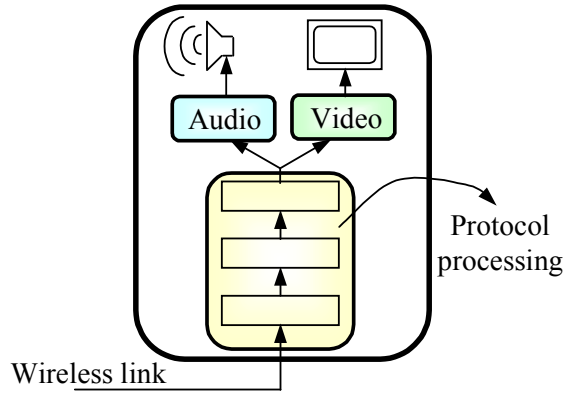Table 2: Task timing parameters for Examples 1 and 2

Figure 1: Mobile multimedia terminal used in Examples 1 and 2.

Example 1: Consider a simple mobile multimedia terminal shown in Figure 1[1]. The system receives real-time audio and video streams over a wireless link, and plays them out. Thus, the three main tasks that run on a processor embedded in this system are protocol processing, audio decoding, and video decoding. The timing parameters for these tasks are listed in Table 2. For simplicity, the initial phase offsets for the tasks are set to zero (however, our algorithm, presented in Section 5, makes no such assumptions). The resulting schedule for the time interval [0, 120] when this task set is scheduled on a single processor using the RM priority assignment scheme, is shown in Figure 2(a). It can be seen from the gure that the system is idle during time interval [90, 120]. This slack can be utilized to lower the operating frequency and supply voltage, thereby reducing the energy consumption. Figure 2(b) shows the schedule for the same task set with the processor slowed down[2] by a factor of $\frac{4}{3}$. As seen from the gure, processor slowdown leads to a reduction in slack[3]. Note that any further reduction in processor speed will result in the video decoding task missing its deadline. When the supply voltage is also scaled, this leads to a decrease in power consumption from 420mW to 184mW, using the power *vs.* frequency curve for the StrongARM processor, shown in Figure 4. The energy consumption over the interval [0, 120], therefore, decreases by 41%, compared to a shutdown based policy. ∎

2.2.2 Dynamic slack

Static slack is not the only kind of slack present in the system. During runtime, due to the variation in the task instance execution times, there is an additional slack created when a task instance nishes executing before its WCET. This slack that arises due to execution time variation is henceforth referred to as **dynamic slack**. Dynamic slack can be exploited for DVS by dynamically varying the operating frequency and supply voltage of the processor to extend the task execution time to its WCET. Thus, the lower a task instance's actual execution time, the more its energy reduction potential. However, to realize this potential, we need to know/estimate the execution time of each

---

[1] The system shown in Figure 1 is only an illustrative example of a multi-tasking, uniprocessor, battery powered embedded system. This could, in general, be a 3G cell phone or a handheld PC.

[2] For simplicity, we assume that such an exact slowdown is possible. Our algorithm handles the case where the processor only has a nite set of available frequencies

[3] While in this example, uniform slowdown of all tasks resulted in a complete elimination of static slack, in general this might not be the case, and different tasks might require different slowdown factors. Our algorithm does this, if necessary.
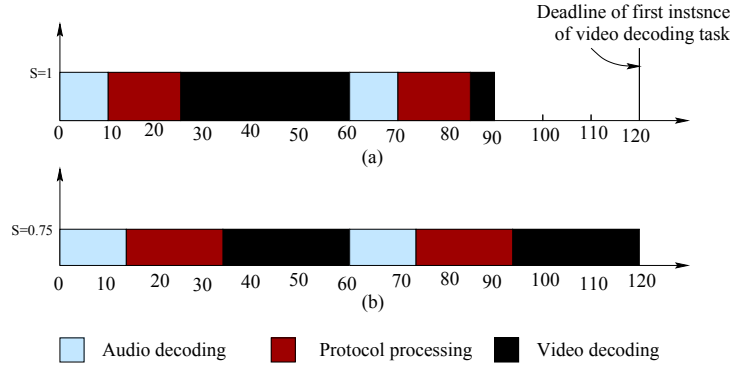
Figure 2: Task execution schedule for Example 1: (a) Original (b) Statically optimized

task instance. The following example illustrates how dynamic slack can be exploited for DVS.

Example 2: Consider the statically optimized schedule for the task set of Example 1 shown in Figure 2(b). Figure 3(a) shows the resulting schedule when the video decoding task instance requires only 20 time units to complete execution at maximum processor speed. As a result, the video decoding task now completes execution at time $t = 60$ units, and does not get preempted. As seen from the gure, this execution time variation creates some dynamic slack. To utilize this slack, the processor frequency and supply voltage are dynamically reduced further during the execution of the video decoding task. If the frequency were to be dropped by a factor of 2 over the already statically optimized value, the slack gets lled up again as shown in Figure 3(b)[4]. Accompanied by appropriate voltage scaling, the energy consumption for the interval [0, 120] now reduces by a further 14% compared to the statically optimized schedule of Figure 3(a). ∎
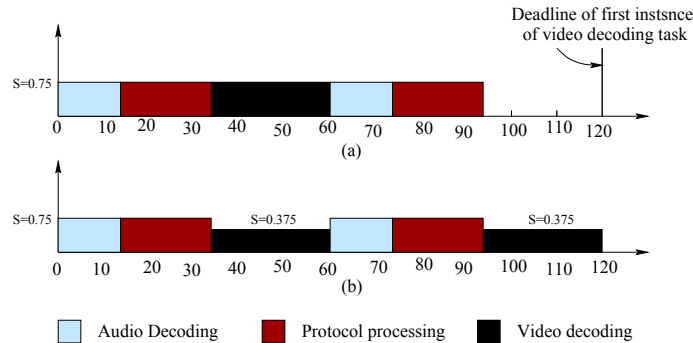
Figure 3: Task execution schedule for Example 2: (a) Statically optimized (b) After dynamic slowdown

The above examples illustrated the energy reduction opportunities present during real-time task scheduling. We next describe our system model, and present an enhanced RM schedulability analysis for variable voltage systems. Using this analysis, we then present our power aware scheduling algorithm that exploits the above illustrated DVS/DPM opportunities to yield large energy savings.

---

[4]Note that in Figure 3(b), the video decoding task is now preempted at time $t = 60$ units, and resumes execution later to complete at time $t = 120$ units.

3 SYSTEM MODEL

3.1 Task timing model

A set of N independent periodic tasks is to be scheduled on a uniprocessor system. Associated with each task *i* are the following parameters: (i) T_i is its time period, (ii) C_i is its worst case execution time (WCET), (iii) B_i is the best case execution time (BCET), and (iv) D_i is its deadline. The BCET and WCET can be obtained through execution pro ling or other performance analysis techniques [15]. Our techniques are applicable to the general case where D_i and T_i are unrelated. We discuss this further in Section 4.

3.2 Power model

We use a power model of the StrongARM processor, which is based on actual measurements, for computing the power consumption. Figure 4 shows the power consumption of the StrongARM, plotted as a function of the operating frequency. This plot is obtained from actual current and voltage measurements reported in [36] for the StrongARM SA-1100 processor. Using this curve, the power consumption of the system for a given clock frequency can be
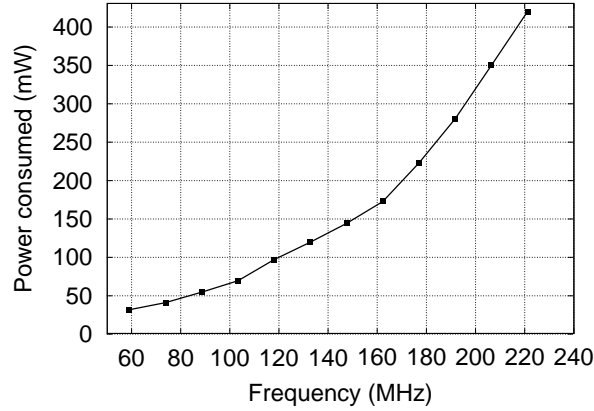
Figure 4: Power consumption as a function of clock frequency for the StrongARM processor

calculated. By varying the supply voltage and clock frequency, the variable voltage system can be made to operate at different points along this curve. For a given clock frequency (*i.e.,* speed setting), the energy consumption of a task instance can be computed as the product of the power consumption (obtained from the power-frequency curve shown above) and the execution time of the task instance. Further, it is easy to see that the energy-speed curve is convex in nature [13]. Thus, if two task instances have to be completed by a deadline, due to Jensen's inequality $\left(\overline{E(r)} \geq E(\overline{r}) \right)$ [37], it is more energy ef cient to run the two tasks at a constant speed than to change the speed from one task to the other. As will be seen in the next section, our algorithm utilizes this fact by attempting to slow down tasks in a uniform manner, to the extent possible.

3.3 Overheads due to DVS/DPM

The variable supply voltage is generated using DC-DC switching regulators. During a voltage transition, it takes a nite amount of time for the output voltage of the DC-DC converter to transition from the present level to the

desired level. Ef cient DC-DC regulators with fast transition times were reported in [38, 39]. Complementary to the supply voltage variation is the accompanying variation of the clock frequency. The phase locked loop present in the clock generation circuitry also takes a nite amount of time to settle at its steady state value, after a frequency change. Further, shutting down and waking up the processor involve a non-zero time and power overhead. These overheads have to be explicitly accounted for during task scheduling. For variable voltage real-time systems, the timing overheads have to be incorporated into the task-set schedulability analysis to guarantee the schedulability of tasks. The energy overheads have to be considered while computing the energy savings obtained through the use of DVS and DPM. Commercial processors till recently had large frequency transition overheads (for example, the StrongARM [20] takes 150 μseconds to change its frequency). However, these transition times are considerably lower in more recent processors as designers realize the effectiveness of DVS. The transition overhead is around 30 μseconds in the XScale processor [17]. We use a stall duration of 150 μseconds in our simulations, since the power model used is that of a StrongARM processor. However, in our implementation (Section 7), the stall duration is only 30μseconds, since our experimental testbed is based on the XScale processor. Finally, although some processors [23] developed at academic institutions allow the computation to continue while the voltage and frequency are being changed, neither the StrongARM nor the XScale processors permit this. Therefore, in our simulations, the processor is stalled for the duration of the frequency change, resulting in some wasted energy (although this is insigni cant since the processor clock is frozen). We account for all the above mentioned overheads in our energy calculations.

4 RATE-MONOTONIC SCHEDULABILITY ANALYSIS

Several classical results exist on the schedulability analysis for RM scheduling [40]. However, none of them consider variable voltage/frequency processors. Since processor shutdown, processor wakeup, and voltage and frequency changes take a nite amount of time, they affect the timing behavior of the system. In this section, we present an enhanced schedulability analysis for RM scheduling, which takes into account the overheads introduced due to DPM/DVS. We denote the time taken to shutdown/wakeup the processor by T_S, and the time taken for a voltage/frequency change by T_V[5].

4.1 The $D_i \leq T_i$ case

The schedulability analysis for RM scheduling for the case $D_i \leq T_i$ is presented in [40], in which necessary and suf cient conditions are derived for the schedulability of a non-concrete (*i.e.*, unknown initial phases for the tasks) periodic task set. The *response time* of a task instance is de ned as the amount of time needed for the task instance to nish execution, from the instant at which it arrived in the system. The worst case response time (WCRT), as the name indicates, is the maximum possible response time that a task instance can have. For a conventional x ed speed system, the WCRT of a task under the RM scheduling scheme is given by smallest solution of the equation [40]:

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j \tag{1}$$

---

[5]The time taken for a voltage change could be different from the time taken for a frequency change. In that case, T_V is the maximum of the two values. Further, the voltage/frequency change will be accompanied by a context switch, whose overhead can also be added to T_V.

where $hp(i)$ denotes the set of tasks with priority greater than that of task i, C_i denotes the worst case execution time of task i, $\lceil \cdot \rceil$ denotes the ceiling operator, and R_i is simply an intermediate variable used in computing the WCRT. Equation (1) can be solved using an iterative technique. The summation term on the right-hand-side of the equation represents the total interference that an instance of task i sees from higher priority tasks during its execution. The non-concrete task set schedulable iff the WCRT of the task is less than or equal to its deadline D_i. Our enhanced schedulability analysis is based upon three observations.

Observation 1: Slowing down a task by a factor α increases its computation time by the same factor. Therefore, given a set of slowdown factors α_i, $i \in \{1, \ldots, N\}$, the computation time for task i, now becomes $(\alpha_i \cdot C_i)$.

Observation 2: Each higher priority task instance that arrives during the lifetime of a lower priority task instance adds either T_V, or $2 \cdot T_V$ to the response time of the lower-priority task instance, where T_V is the time taken to change the processor frequency/voltage. This is the preemption related overhead of DVS.

It takes T_V time units to change the voltage to execute the higher priority task instance, and another T_V units to change back to resume execution of the preempted task instance, and the worst case for a task instance occurs when this procedure repeats for every higher priority task instance that arrives during its lifetime. So, the worst case interference that a higher priority task instance contributes is equal to $2 \cdot T_V$. However, when the higher priority task is being executed, suppose a task with priority in between that of the currently running task and the preempted task arrives. It is easy to see that this task instance only causes an interference of T_V.

Observation 3: The maximum amount of blocking that can be faced by a task instance, and which is not preemption related is $Max. \{2 \cdot T_{shut} + T_V, \ 2 \cdot T_V\}$, where T_{shut} is the time taken to shutdown/wakeup the processor.

Since the processes of voltage/frequency change or shutdown cannot be preempted, each task instance may be blocked for an amount of time if it arrives just after a voltage/frequency change or a shutdown has been initiated. Based on the above observations, the following theorem can be used as a suf cient condition for the schedulability of a task set under RM scheduling on a DVS enabled system.

Theorem 1 (Suf cient RM schedulability condition): *A non-concrete periodic task set is guaranteed to be schedulable on a variable voltage processor, under the RM scheduling policy if, for every task i, $WCRT(i) \leq D_i$ where $WCRT(i)$ is the smallest solution of the equation:*

$$R_i = \alpha_i \cdot C_i \ + \ Max. \{2 \cdot T_S + T_V, 2 \cdot T_V\} \ + \ \sum_{j \in hp(i)} \left(\left\lceil \frac{R_i}{T_j} \right\rceil \times (\alpha_j \cdot C_j + 2 \cdot T_V) \right) \tag{2}$$

where α_n is the slowdown factor for task n.

4.2   The $D_i > T_i$ case

The schedulability analysis for the case when $D_i > T_i$ is more complex. For x ed speed systems, when D_i and T_i are unrelated, the WCRT of task i is given by [40]:

$$WCRT(i) = Max._{\ 0 \leq q \leq Q} \ \{W_{i,q} - q \cdot T_i\} \tag{3}$$

where Q is the smallest non-negative integer value that satis es $W_{i,Q} \leq (Q+1) \cdot T_i$, and $W_{i,q}$ is the smallest solution to the equation:

$$W_{i,q} = (q+1) \times C_i + \sum_{j \in hp(i)} \left\lceil \frac{W_{i,q}}{T_j} \right\rceil \times C_j \qquad (4)$$

The task set is schedulable if, and only if, $WCRT(i) \leq D_i$, $\forall i \in \{1, \ldots, n\}$. Intuitively, the above analysis implies that the WCRT may not occur for the rst instance of the task, and we may need to check the schedulability for multiple instances of the task. In the case of variable voltage systems, using the three observations listed in Section 3.4, the schedulability test reduces to a suf cient condition, and Equation (4) is changed to:

$$W_{i,q} = (q+1) \times \alpha_i \times C_i + Max. \{2 \cdot T_S + T_V, 2 \cdot T_V\} + \sum_{j \in hp(i)} \left(\left\lceil \frac{W_{i,q}}{T_j} \right\rceil \times (\alpha_j \cdot C_j + 2 \cdot T_V) \right) \qquad (5)$$

where α_i is the slowdown factor for task i. It is worth mentioning that the $D_i \leq T_i$ case is just a special case of this enhanced equation. When $D_i \leq T_i$, Equations (3) and (4) are satis ed for $Q = 0$. The of ine component of our algorithm uses this enhanced suf cient schedulability test to compute static slowdown factors for each task.

5   ALGORITHM

We consider a priority based, preemptive scheduling model where the priority assignment is either done statically, in the case of Rate Monotonic scheduling[6]. Task instances that do not nish executing by their deadline are killed. Our algorithm adjusts the processor's supply voltage and clock frequency whenever a new task instance rst starts executing, as well as every time it resumes execution after being preempted by a higher priority task. Also, our algorithm uses a constant speed setting between two points of preemption in order to maximally exploit the convexity of the energy-speed curve. The pseudo-code of our algorithm is shown in Figures 5 and 6. The crucial steps of our algorithm are described next.

5.1   Static slowdown factor computation

This component of our algorithm involves a schedulability analysis of the task set. It is executed whenever a new task (*e.g.*, audio/video decoding) enters the system and registers itself with the scheduler. Given the task set to be scheduled, the procedure COMPUTE_STATIC_SLOWDOWN_FACTORS performs a schedulability analysis (described in Section 3), and computes the minimum operating frequency for each task, at which the entire task set is still schedulable. Every task is slowed down uniformly till one or more tasks reach their critical point (*i.e.*, they are "just" schedulable). This is done by the function Scale_WCET(). At this juncture, further slowdown of any task with higher priority than any of the critical tasks will render at least one critical task unschedulable. Therefore, only the tasks, if any, with lower priority than any of the critical tasks can be slowed down further without affecting the schedulability of the task set. The procedure continues till there is no further scaling possible while satisfying all task deadlines. In summary, this iterative procedure gives us a new reduced frequency for each task such that the

---

[6]Although we do not analyze it in this paper, our DVS algorithm is also applicable to dynamic priority scheduling, such as Earliest Deadline First, using the appropriate schedulability analysis.

```
Procedure COMPUTE_STATIC_SLOWDOWN_FACTORS
Inputs: Time_Periods[ ], WCETs[ ], Deadlines[ ]
Outputs: Static_Slowdown_Factors[ ]
{
SET S = τ          //Tasks that can be slowed down further
SET S1 = φ         //Tasks that will miss deadline upon further scaling
Current_Scaling_Factor = 1;
For each task i in S
  Static_Slowdown_Factor[i] = 1;
While (S != φ) {
  f = Scale_WCET (Time_Periods[ ], WCETs[ ], Deadlines[ ], S, Static_Slowdown_Factor[ ]);
  S1 = Tasks that will miss deadlines with further scaling;
  Current_Scaling_Factor *= f;
  For each task i in S
    Static_Slowdown_Factor[i] = Current_Scaling_Factor;
  S = All tasks with priority < Lowest priority of tasks in S1;
  }
}
```

Figure 5: Pseudo-code for the offine component of the proposed algorithm

entire task set is just schedulable. The online component of our algorithm augments this static slowdown with a dynamic slowdown factor, computed at runtime, in order to increase energy savings.

5.2 Dynamic slowdown factor computation

The online component of our algorithm invokes the COMPUTE_DYNAMIC_SLOWDOWN_FACTOR procedure, listed in Figure 6, to dynamically alter the supply voltage and operating frequency of the system in accordance with recent task execution statistics. Thus, while the offine component of our algorithm computes *task* specific static slowdown factors, the online component augments this with *task-instance* specific dynamic slowdown factors. The COMPUTE_DYNAMIC_SLOWDOWN_FACTOR procedure is called each time a new task instance starts, or resumes execution after being preempted by a high priority task. The algorithm sets the processor speed and voltage based on an estimate of the task instance execution time.

5.2.1 Runtime prediction strategy

A novel feature of our algorithm is that it involves a *proactive* DVS scheme. In contrast, most existing techniques are reactive in nature. Therefore, in conventional schemes, any slack that arises due to a task instance completing execution early, is distributed to task instances that follow, using (possibly complex) dynamic slack reclamation algorithms. Our technique, on the other hand, attempts to avoid the creation of slack altogether, through the use of a predictive technique, eliminating the need for slack reclamation. In our predictive scheme, the execution time of a task instance is predicted as some function of the execution times of a fixed number of previous task instances. An execution history database is maintained for each task, that is updated every time a task instance finishes executing

```
Procedure COMPUTE_DYNAMIC_SLOWDOWN_FACTOR
Inputs: WCET, Deadline_Miss_History, Execution_History
Outputs: Dynamic_Slowdown_Factor
{
P = PREDICT_TIME (Execution_History);
α = ADAPTIVE_FACTOR (Deadline_Miss_History, α);
P = P * α;
Dynamic_Slowdown_Factor = WCET / P;
}

Procedure PREDICT_TIME
Input: Execution_History
Output: Predicted_time
{
If (Elapsed_time_for_task_instance == 0)
```

$$\text{Predicted\_time} = \frac{\sum_{i=1}^{N} (a_i \times Execution\ time\ of\ kth\ task\ instance)}{\sum_{i=1}^{N} a_i};$$

```
  //N is the number of past instances being monitored
  //The coefficients  a_i depend on the prediction scheme used
Else
  Predicted_time = Expected value of execution time
                   distribution from Elapsed_Time to WCET;
}

Procedure ADAPTIVE_FACTOR
Input: Deadline_Miss_History, Adaptive_Factor
Output: Adaptive_Factor
{
x = Number of deadline misses in past WINDOW instances;
If (x ≥ T1) Adaptive_Factor + = I;
If (x ≤ T2) Adaptive_Factor - = D;
If (Adaptive_Factor ≤ ADAPTIVE_LB) Adaptive_Factor = ADAPTIVE_LB;
}
```

Figure 6: Pseudo-code for the online component of the proposed algorithm

or is killed due to a deadline miss. The function used to compute the execution time determines the type of predictor. We have evaluated our techniques using a simple average predictor model, as well as an exponentially weighted average one. Both these are simple prediction schemes that place a light computational load on the scheduler. Since the results were very similar in the two cases, we report only the results obtained using the weighted average model. The use of a more complex predictor will improve prediction accuracy. However, this will increase the computational burden on the scheduler since the predictor is executed every time a task instance starts or restarts after preemption.

5.2.2 Improving prediction accuracy

In order to improve prediction accuracy while retaining simplicity, we extend the prediction strategy to use *conditional prediction* as described below. Every time a task instance is preempted, the predicted remaining time for that task instance is recalculated as the expected value of the past execution history distribution from the already elapsed

time to the WCET of the task. This gives the predicted remaining time of the task, given that it has already run for the elapsed time. This improves the prediction accuracy, and reduces the probability of under-prediction, in turn reducing the probability of missed deadlines. As a side effect, it also reduces the impact of the original prediction model. This explains why we obtained similar results using the simple average and weighted average predictors.

5.2.3   Adaptive power- delity tradeoff

Some applications may not be able to tolerate the number of deadlines that are missed using the above described predictive scheme. Therefore, we introduce an adaptive feedback mechanism into the prediction process. The predicted execution time of a task instance is altered using an adaptive multiplicative factor, as shown in the pseudo code of Figure 6. Deadline miss history is monitored using a moving window mechanism, and if the number of deadline misses is found to be increasing, the prediction is made more conservative, reducing the probability of further deadline misses. Similarly, a low/decreasing deadline miss history results in more aggressive prediction in order to reduce energy consumption. This is implemented as follows. If there are more than $T1$ deadlines misses in the last $WINDOW$ task instances, the algorithm becomes more conservative and increases the adaptive factor by I. Similarly, if the number of deadline misses in the last $WINDOW$ task instances is less than $T2$, then the algorithm becomes more aggressive and decreases the adaptive factor by D. A lower bound, $ADAPTIVE\_LB$, dictates the maximum aggressiveness of the algorithm. The prediction scheme is now adaptive to a recent history of missed deadlines, becoming more conservative in response to deadline misses and becoming more aggressive if no deadline misses occur over the past few instances. This adaptive control mechanism ensures that the number of deadline misses (which is representative of system delity) is kept under tight check at all times. The choice of the various parameters depends on how aggressive/conservative the user wants to be in the energy- delity tradeoff, a detailed analysis of which is beyond the scope of this paper. Note that in order to keep the algorithm and implementation simple, we have made the deadline miss history and all the parameters of the adaptive algorithm global parameters. An alternative, although more complicated, approach would be to perform the adaptation on a per-task basis. Such an approach would permit better control on the delity of individual applications. Finally, note that this adaptive scheme is also useful in the case when applications can tolerate more deadlines than missed through the use of the baseline predictive scheme. In such a situation, the adaptive scheme can be used to obtain further energy savings at the cost of more deadline misses. Thus, the application can adjust the operating point along an energy- delity curve, which enhances its energy scalability. For example, as the system runs out of energy, the it can scale down its delity gracefully. To the best of our knowledge, none of the existing DVS schemes provide this capability.

5.2.4   Voltage variation policy

Our algorithm recomputes the voltage setting every time a task instance rst starts executing, or restarts after being preempted. The pre-computed static slowdown factor for the task is augmented with a dynamic slowdown factor for the speci c task instance that is about to start executing. The dynamic slowdown factor is computed by stretching the task instance's predicted execution time to reach its WCET. The product of the static and dynamic factors is the nal slowdown factor. The processor's operating frequency is then decreased by this factor, the corresponding supply voltage is set, and execution of the task instance begins. This dynamic slowdown spreads the execution to ll
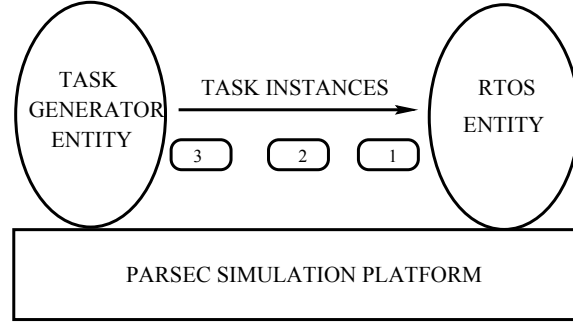
Figure 7: System simulation model in PARSEC

otherwise idle time intervals, enabling operation at a lower supply voltage and clock frequency.

6 SIMULATION BASED PERFORMANCE ANALYSIS

We next describe our simulation framework, and present simulation results comparing the proposed adaptive power- delity DVS algorithm to several existing DVS/DPM schemes to demonstrate its effectiveness.

6.1 Simulation model

To analyze the performance of the proposed adaptive power- delity DVS scheme, a discrete event simulator was built using PARSEC [41], a C based parallel simulation language. The simulation structure, shown in Figure 7, consisted of two parallel communicating entities. The rst one represented the RTOS, and implemented the task scheduler (enhanced with the DVS scheme). In addition to performing task scheduling, the RTOS entity also maintained the statistics of task instance execution times and deadline misses. The second entity, *i.e.*, the task generator, periodically generated task instances with run times according to a trace or a distribution, and passed them to the RTOS entity.

6.2 Simulation results

We performed simulations on two task sets (henceforth called *Task Set 1* and *Task Set 2*) that are based on standard task sets used in embedded real-time applications [42, 43]. The task instance execution times were generated using a Gaussian distribution[7] with $Mean = \frac{BCET+WCET}{2}$, and $Standard\ Deviation = \frac{WCET-BCET}{6}$. To demonstrate that our technique works even for non-concrete task sets, every task i was given a random initial phase offset in the interval $[0, T_i)$. We performed four experiments on each task set. In the rst experiment, only shutdown based DPM was employed, and the supply voltage and frequency were x ed at their maximum values. Next, we implemented the low-power scheduling technique of [14]. This is a WCET based scheme, where DVS is done only when there is a single task remaining to execute. In the third experiment, we implemented the PAST/PEG DVS scheme [18, 19]. Similar to our algorithm, this is a predictive DVS scheme. However, all DVS decisions in the PAST/PEG scheme are purely utilization based, and therefore the PAST/PEG scheme performs poorly in the presence of deadlines. Finally, the proposed adaptive power- delity DVS scheme was implemented. The experiments were repeated while varying

---

[7]We also repeated our experiments using a uniform distribution. The results were similar in both cases. Therefore, we present only the results for the Gaussian distribution case.
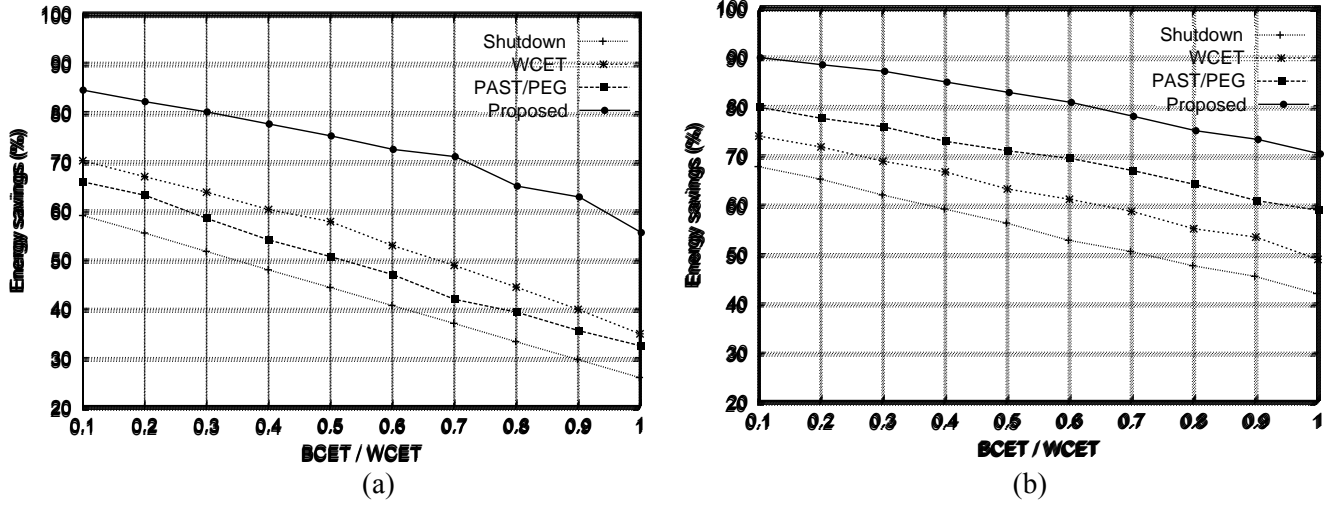
Figure 8: Energy savings of various DVS/DPM schemes under RM scheduling for (a) Task Set 1 and (b) Task Set 2
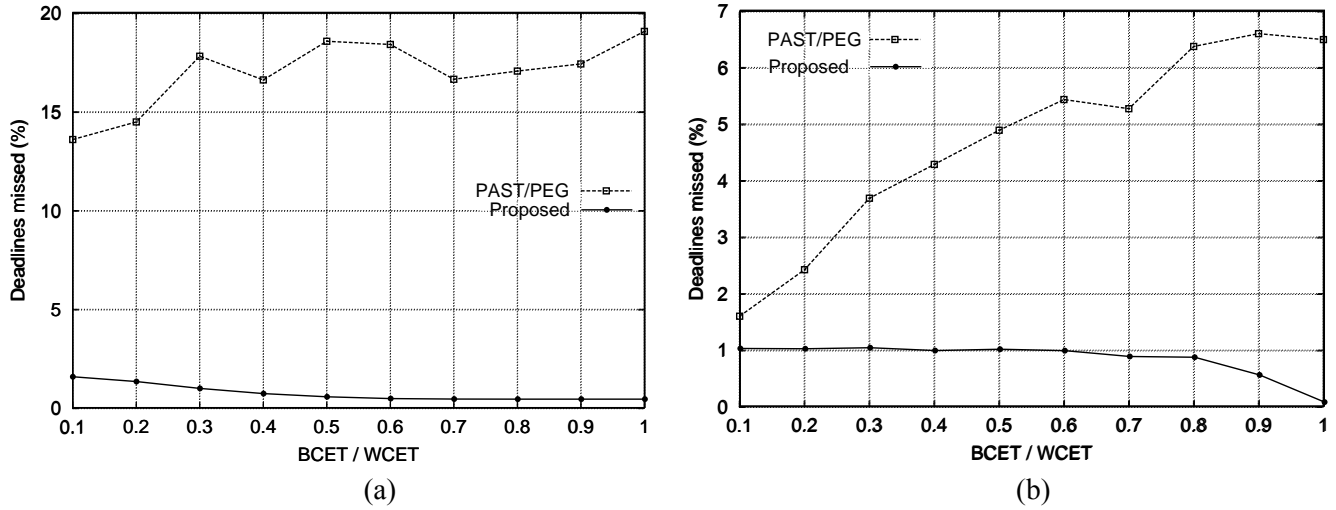
Figure 9: Deadline miss percentage for the proposed scheme and the PAST/PEG scheme under RM scheduling for (a) Task Set 1, and (b) Task Set 2

the BCET from 10% to 100% of the WCET in steps of 10%. Each data point was averaged over 50 simulation runs, each run simulating the execution of 50,000 task instances. For the adaptive scheme, the following parameter values were used: $N = 10$, $WINDOW = 10$, $I = 0.05$, $D = 0.01$, $T1 = 1$, and $T2 = 0$. The various parameters are described in Section 5.2.3.

Figure 8 shows the energy savings obtained for the two task sets under RM scheduling, for each of the above mentioned DVS/DPM schemes. The results are normalized to the case when no DVS/DPM is used. As can be seen in the gure, the proposed adaptive power- delity DVS algorithm results in signi cantly higher energy savings compared to the shutdown, WCET based, and PAST/PEG algorithms. Figure 9 shows the percentage of deadlines missed by the proposed algorithm, and the PAST/PEG algorithm. The PAST/PEG scheme results in a large number of deadline misses because it takes DVS decisions purely based on processor utilization. As mentioned before, in real-time multi-tasking systems, the schedulability of the task set is only weakly related to the processor utilization.

(a)

(b)
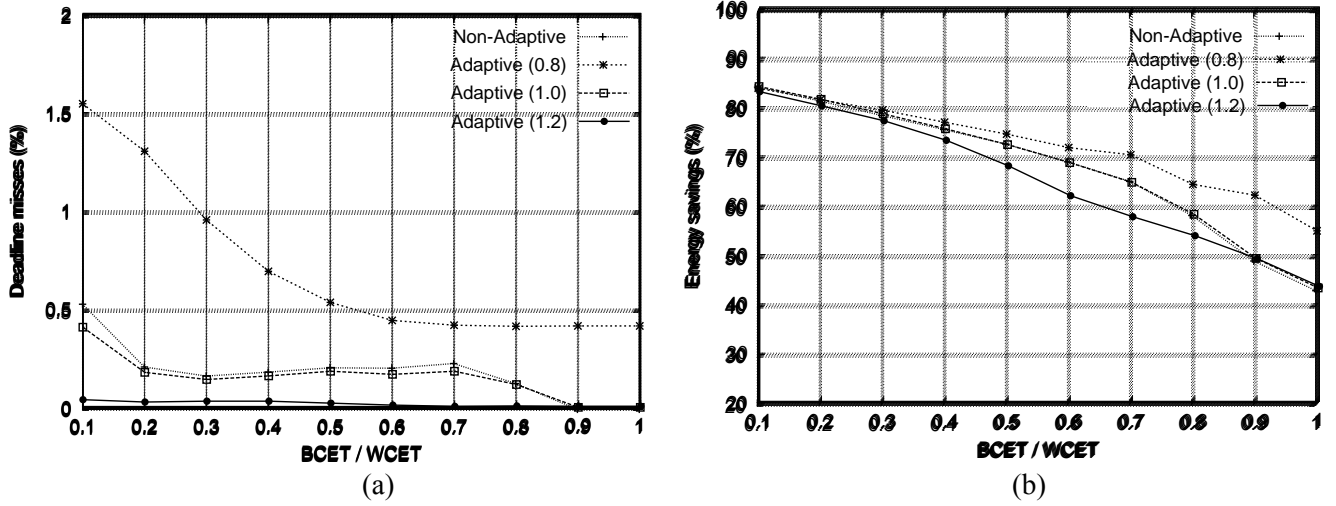
Figure 10: Impact of the parameter $ADAPTIVE\_LB$ on (a) Percentage of deadline misses and (b) Energy savings, for Task Set 1 under RM scheduling

Therefore, the PAST/PEG algorithm results in a signi cant loss in real-time behavior. Note that this is in contrast to what was observed by Farkas *et al.* [19]. That is because they only considered a unitasking environment where the schedulability is determined completely by processor utilization. The algorithm proposed in this work is tightly coupled to the schedulability analysis of the underlying real-time scheduling scheme used, resulting in far fewer deadline misses, as is evident from Figure 9.

(a)

(b)
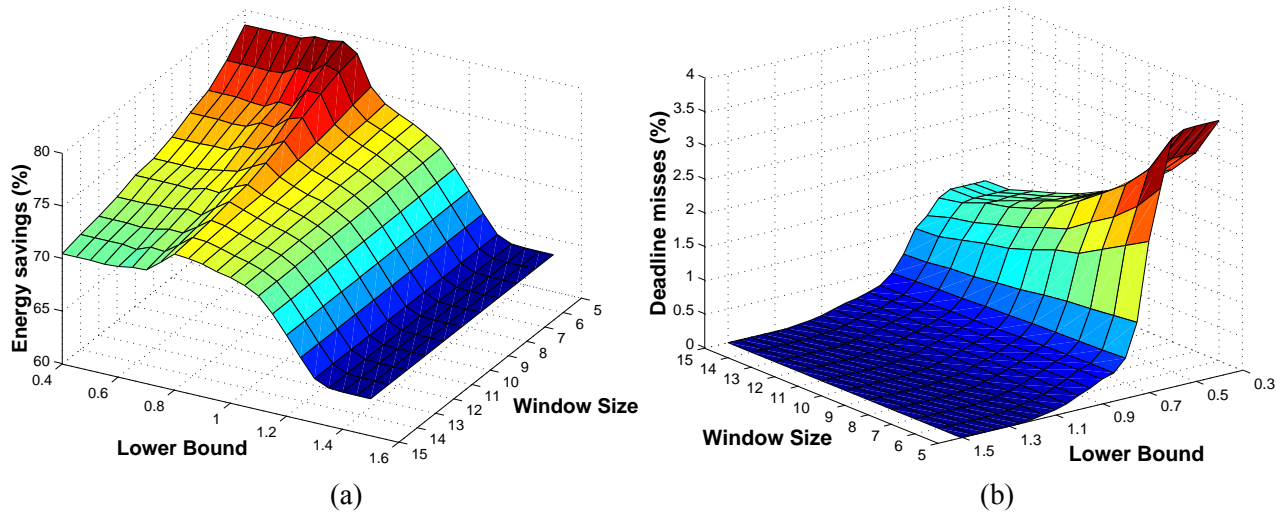
Figure 11: Variation of (a) Energy savings and (b) Deadline miss percentage with parameters $WINDOW$ and $ADAPTIVE\_LB$ for Task Set 1 under RM scheduling

We performed additional experiments with Task Set 1 to illustrate the adaptive power- delity tradeoff that our scheme provides. We ran our algorithm for three different values of the parameter $ADAPTIVE\_LB$. Figures 10(a) and 10(b) show the energy savings and deadline miss percentages, respectively, for $ADAPTIVE\_LB \in \{0.8, 1.0, 1.2\}$, and for a non-adaptive version of our algorithm. From the gures, it is clear that by increasing $ADAPTIVE\_LB$, one can trade-off energy savings for fewer deadline misses. Such an adaptive scheme enhances the system's energy-
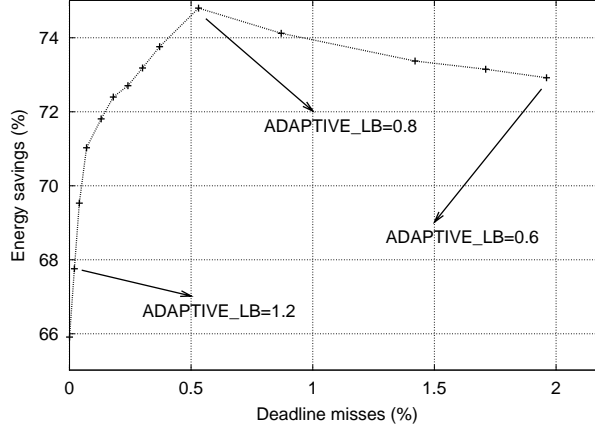
Figure 12: Energy savings *vs.* Deadline misses for $WINDOW = 10$ and various values of $ADAPTIVE\_LB$ for Task Set 1 under RM scheduling.

scalability. For example, as the battery drains out, $ADAPTIVE\_LB$ can be decreased, thereby gradually degrading the system's delity . Figures 11(a) and 11(b) plot the energy savings and deadline misses as a function of the parameters $ADAPTIVE\_LB$ and $WINDOW$, for a $\frac{BCET}{WCET} = 0.5$. It is evident that as $ADAPTIVE\_LB$ increases, the energy savings decrease, and the deadline misses increase. As the window size increases, the energy savings decrease and the deadline misses decrease. Also, note that choosing a very low value for $ADAPTIVE\_LB$ is not very energy ef cient. This is because, the large number of missed deadlines cause the voltage to be increased very frequently in response, lowering the energy savings obtained. In order to clearly show the energy- delity tradeoff, we have plotted the energy savings *vs.* the deadline miss percentage in Figure 12 for $WINDOW = 10$ and different values of $ADAPTIVE\_LB$. As can be seen in Figure 12, a value of 0.8 for $ADAPTIVE_{L}B$ seems to yield the maximum energy savings, for this particular task set.

7   IMPLEMENTATION

In addition to validating the proposed adaptive power- delity technique through simulations, we also evaluated its performance by implementing it into an RTOS running on a complete variable voltage hardware platform. In this section, we rst give a brief overview of a structured software architecture that we have developed to facilitate application-RTOS interaction for effective energy management. Then, we describe our implementation in detail, and present measured results that demonstrate the effectiveness of our DVS algorithms.

7.1   Software architecture

We view the notion of power awareness in the application and OS as a capability that enables a continuous dialog between the application, the OS, and the underlying hardware. This dialog establishes the functionality and performance expectations (or even contracts, as in the real-time sense) within the available energy constraints. A well structured software architecture is necessary for realizing this notion of power awareness. The power aware software architecture (PASA) [44, 45] that we have developed is composed of two software layers and the RTOS kernel. One layer is an API that interfaces applications with the OS, and the second layer makes power related hardware knobs

199
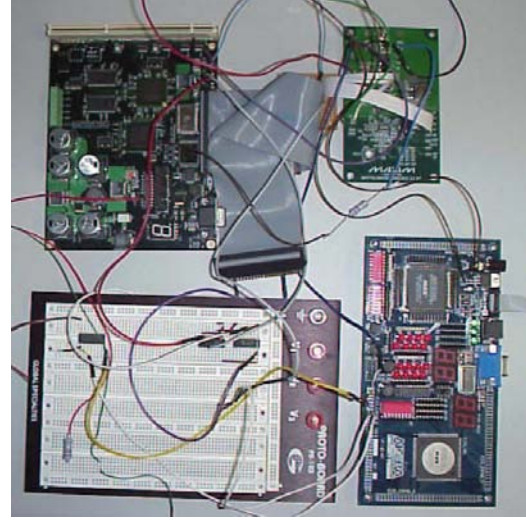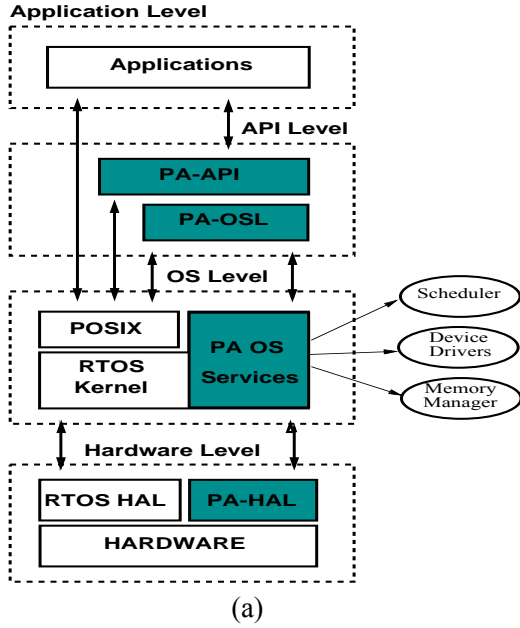
Figure 13: (a) Power Aware Software Architecture, and (b) Picture of the experimental setup. The XScale board is on the upper left side, the Maxim board on the upper right side, and the FPGA board, which generates interrupts, is on the lower right side. On the lower left side is a bread board with interconnections to trigger the DAQ board.

available to the OS. Both layers interface to various OS services as shown in Figure 13(a). The API layer is separated into two sub-layers. The Power Aware Application Programmer Interface (PA-API) sub-layer provides power management functions to the applications, while the other sub-layer, the Power Aware Operating System Layer (PA-OSL) provides access to existing and modi ed OS services. Active entities that are not implemented within the RTOS kernel should be implemented at this layer (*e.g.*, threads created to assist the OS in DVS/DPM, such as a thread responsible for killing other threads whose deadlines were missed). The modi ed RTOS and the underlying hardware are interfaced using a Power Aware Hardware Abstraction Layer (PA-HAL). The PA-HAL gives the OS access to the power related hardware knobs while abstracting out the details of the underlying hardware platform.

7.2 Experimental setup

Using the PASA presented above, the proposed adaptive power- delity DVS algorithm has been incorporated into the *eCos* operating system, an open source RTOS from Red-Hat Inc. *eCos* was ported to an 80200 Intel Evaluation Board[46], which is an evaluation platform based on the XScale processor. The XScale processor supports nine frequency levels ranging from 200MHz to 733MHz. However, two of them (200MHz and 266MHz) cannot be used in the 80200 board due to limitations in the clock generation circuitry [17]. In addition, the processor supports three different low power modes: IDLE, DROWSY, and SLEEP. The SLEEP mode results in maximal power savings, but requires a processor reset in order to return to the ACTIVE mode. The IDLE mode, on the other hand, offers the least power savings but only requires a simple external interrupt to wake the processor up. We use the IDLE power down mode in our experiments due to its simple implementation.

Like most RTOSs, *eCos* requires a periodic interrupt to keep track of the internal OS tick, responsible for the notion of timing within the system. In the 80200 board, the only source of such an interrupt is the internal XScale

Clock Frequency (MHz)	Supply Voltage (V)
733	1.5
666	1.4
600	1.3
533	1.25
466	1.2
400	1.1
333	1.0

Table 3: Frequency-Voltage pairs for the Intel XScale processor used in our implementation

performance counter interrupt. However, the interrupt is internal to the processor, and therefore cannot wake it up from the IDLE mode. To overcome this problem, we used a source of external interrupts to wake up the processor. The interrupt pin of the processor is connected to an Altera FPGA board, which generates periodic interrupts to wake up the processor. The period of the external interrupt is made equal to that of the smallest time period task to ensure that the processor is not woken up unnecessarily. The experimental setup, consisting of the XScale board, the MAXIM DC-DC converter board, the FPGA, and some interface circuitry is shown in Figure 13(b).

7.2.1 Variable voltage supply

The variable voltage supply consists of a MAXIM 1855 DC-DC converter board, and some interface circuitry implemented using a PLD. When a voltage change is required, the processor sends a byte through the peripheral bus of the 80200 board to the interface circuitry which acts as an addressable latch. The outputs of this latch are connected to the digital inputs of the Maxim variable supply board. These inputs determine the output supply voltage of the Maxim board, which is fed back to the processor core. For the experiments, the system was con gured to run at supply voltages from 1.0V to 1.5V and corresponding frequencies from 333 MHz to 733 MHz. The frequency-voltage pairs are listed in Table 3. The frequency is changed using the XScale's internal registers.

7.2.2 Power measurement setup

We used a National Instruments Data Acquisition (DAQ) board to measure the power consumption. We isolated the power supply to the processor from the rest of the board, and measured the power consumed by the processor alone. For this purpose, we used a shunt resistor (0.02 ohm) and sampled the voltage drop across it at a high sampling rate to obtain the instantaneous current drawn by the processor. We computed the energy consumption by integrating the product of the instantaneous supply voltage and current consumption over the interval of interest. Our measurement setup is similar to the one described in [19]. The DAQ board was triggered by pulling signals out of the peripheral bus of the 80200 board to synchronize the power measurements with the task execution.

7.3 Experimental procedure and results

We implemented and compared three different DVS/DPM schemes, (i) shutdown based DPM, (ii) a non-adaptive version of the proposed algorithm, and (iii) the complete adaptive power- delity tradeoff DVS algorithm. As a baseline for comparison, we also conducted an experiment without any DVS/DPM. The following parameter values were used for the adaptive scheme, $T_1 = 2$, $T_2 = 0$, $WINDOW = 10$, $I = 0.1$, and $D = 0.05$. We used two

Task	Application	Worst case execution time at max. frequency (μs)	Std. Dev. (μs)
T1	MPEG2 (wg_gdo_1.mpg)	30700	3100
T2	MPEG2 (wg_cs_1.mpg)	26300	2100
T3	ADPCM	9300	3300
T4	FFT	15900	0
T5	FFT (Gaussian dist.)	13600	800

Table 4: Applications used in the experiments. T1 and T2 both perform MPEG decoding, but on different les.

Task Set	Component Task	C	T	D	Static Slowdown Factor
A	T2	26300	40000	40000	0.9495
	T3	9300	80000	80000	0.9495
	T4	15900	120000	120000	0.9495
B	T1	30700	47000	47000	0.8979
	T3	9300	94000	94000	0.8979
	T4	15900	141000	141000	0.8979
C	T1	30700	45000	45000	0.9207
	T3	9300	90000	90000	0.9207
	T5	13600	135000	135000	0.9207

Table 5: Task Sets used in the experiments. Each task set is comprised of three tasks. C stands for the worst case execution time, T denotes the time period, and D denotes the deadline. All times are in μseconds.

different values (0.95 and 0.85) for $ADAPTIVE\_LB$. We did not explore optimal choices for these parameters. Rather, our goal was to demonstrate the effectiveness of our algorithm by implementing it on a real system.

We used three different applications in our experiments: (i) an MPEG2 decoder, (ii) an ADPCM speech encoder, and (iii) a oating point FFT algorithm. Note that these applications run concurrently in the system. Each application executes as an eCos thread, and is scheduled using the RM priority assignment scheme. We pro led the execution time of each application for different input data to collect WCET statistics. For the MPEG2 decoder, different les were decompressed, and the WCET was measured separately for each one of them. The original FFT algorithm computes a x ed number (1024) of FFTs in one execution. In order to increase the variability in execution time, we also implemented a version of the FFT algorithm that computes a random number (obtained using a Gaussian distribution) of FFTs in each execution. Table 4 shows the characteristics of the applications used. The WCET of each application instance was measured with the processor running at maximum frequency. The standard deviation is also given in Table 4 to show the variability in the execution times for each application. We built three different task sets using these applications. The task sets, their component tasks, and the individual task characteristics are listed in Table 5. The characteristics indicate the WCET, time period, and deadline, respectively. The static slowdown factor is also listed for each task.

All the DVS/DPM algorithms shutdown the processor as soon as it becomes idle. The processor is woken up when the next external interrupt arrives. A limitation of the processor wake up strategy for our current testbed requires us to make all task periods a multiple of the highest-rate task. Thus, whenever the processor is woken up there is useful work to be done. This limitation could be eliminated through the use of a programmable interrupt generator. Further, note that this is not a limitation of the DVS algorithm or the software architecture itself. The static slowdown factors used by the proposed DVS algorithm are maintained in a table that is internal to *eCos*. Each

DVS/DPM scheme used	Energy (Joules)	Power (Watts)	Ratio	No. of deadlines missed (T2/T3/T4)
No DVS/DPM	39.085	0.779	1	0/0/0
Shutdown	31.504	0.628	0.80	0/0/0
Non-adaptive	28.496	0.568	0.72	1/1/2
Adaptive (0.95)	26.581	0.527	0.68	3/2/1
Adaptive (0.85)	25.251	0.502	0.64	3/1/4

Table 6: Energy and average power consumption for Task Set A. The total number of task instances is 415,207, and 138 for tasks T2, T3, and T4 respectively. For the adaptive schemes, the number in parentheses denotes the value of $ADAPTIVE\_LB$.

DVS/DPM scheme used	Energy (Joules)	Avg. Power (Watts)	Ratio	No. of deadlines missed (T1/T3/T4)
No DPM	12.546	0.798	1	0/0/0
Shutdown	11.265	0.716	0.89	0/0/0
Non-adaptive	9.811	0.624	0.78	1/0/1
Adaptive (0.95)	9.795	0.623	0.78	1/0/1
Adaptive (0.85)	8.828	0.562	0.70	1/1/31

Table 7: Energy and average power consumption for Task Set B. The total number of task instances is 130,65, and 43 for tasks T1, T3, and T4, respectively. For the adaptive schemes, the number in parentheses denotes the value of $ADAPTIVE\_LB$.

task type is associated with a static factor, which is computed during system initialization. The dynamic and adaptive factors are maintained in a table of task instances. The task type table also contains a specied number (10 in our case) of execution times of previous task instances. The *eCos* kernel has full access to these tables. When a context switch occurs, the various tables are updated, and the voltage and frequency of the processor are adjusted.

The energy and average power consumption for the three task sets, under various DPM/DVS schemes, are shown in Tables 6, 7, and 8. The column titled *Ratio* gives the energy consumption normalized to the case when no DVS/DPM is used, and the column titled *Dead. missed* gives the number of missed deadlines for each task. For the adaptive algorithm, the number in parentheses represents the value of $ADAPTIVE\_LB$. As expected, the energy savings increase as this parameter decreases, at the cost of more deadline misses. These results indicate that the proposed adaptive power- delity DVS algorithm does indeed result in considerable energy savings at the cost of a small number of missed deadlines.

8 CONCLUSIONS

This paper presented an RTOS directed DVS scheme to reduce energy consumption in wireless embedded systems. A key feature of the proposed technique is that it yields an adaptive tradeoff between energy consumption and system delity. The proposed algorithm exploits low processor utilization, instance to instance variation in task execution times, and tolerance to missed deadlines of wireless systems to achieve this tradeoff. The technique has been incorporated into the kernel of the eCos RTOS, and an energy efcient software architecture has been developed that facilitates *application aware* power management by enabling a dialog between the application and the RTOS. Involving the application in DVS/DPM can yield signicant benets. In many cases, the execution time

DPM Scheme Used	Energy (Joules)	Avg. Power (Watts)	Ratio	No. of deadlines missed (T1/T3/T5)
No DPM	13.080	0.838	1	0/0/0
Shutdown	12.342	0.772	0.94	0/0/0
Non-adaptive	10.892	0.693	0.83	0/1/18
Adaptive (0.95)	10.958	0.697	0.83	0/1/18
Adaptive (0.85)	9.990	0.637	0.76	11/16/32

Table 8: Energy and average power consumption for Task Set C. The total number of task instances is 130, 65, and 43 for tasks T1, T3, and T5, respectively. For the adaptive schemes, the number in parentheses denotes the value of $ADAPTIVE\_LB$.

is a superposition of several distinct distributions corresponding to different operating modes or distinct values of data. For example, a multiplier takes lesser time to compute its output if the operands are powers of two. Another example is an MPEG decoder whose histogram of run times has three distinct peaks corresponding to P, I and F frames. In such cases, an intelligent task can provide feedback to the OS in the form of a hint on the distribution of run times after the data values are known. As part of future work, we plan to investigate the energy reduction potential of such interactions in detail.

References

[1] A. P. Chandrakasan and R. W. Brodersen, *Low Power CMOS Digital Design*. Kluwer Academic Publishers, Norwell, MA, 1996.

[2] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, Norwell, MA, 1998.

[3] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, Norwell, MA, 1997.

[4] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen, "Predictive system shutdown and other architectural techniques for energy ef cient programmable computation", in *IEEE Trans. on VLSI Systems*, March, 1996.

[5] C. Hwang and A. Hu, "A predictive system shutdown method for energy saving of event driven computation", in *Proc. IEEE ICCAD*, pp. 28–32, November, 1997.

[6] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Dynamic power management for portable systems", in *Proc. ACM MOBICOM*, August, 2000.

[7] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management", in *IEEE Trans. on VLSI Systems*, vol. 8, iss. 3, pp. 299–316, June, 2000.

[8] L. S. Nielsen and J. Sparso, "Low-power operation using self-timed circuits and adaptive scaling of supply voltage", in *Proc. Int. Wkshp. Low Power Design*, pp. 99–104, April, 1994.

[9] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy", in *Proc. First Symp. on OSDI*, pp.13–23, November, 1994.

[10] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU", in *Proc. ACM MOBI-COM*, pp. 13–25, November 1995.

[11] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy", in *Proc. Annual Symp. on Foundations of Computer Science*, pp.374–382, October, 1995.

[12] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processors", in *Proc. IEEE ICCAD*, pp.653–656, November, 1998.

[13] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors, in *Proc. ACM ISLPED*, pp.197–202, August, 1998.

[14] Y. Shin and K. Choi, "Power conscious x ed priority scheduling for hard real-time systems", in *Proc. IEEE/ACM DAC*, pp. 134–139, June, 1999.

[15] Y. -T. S. Li and S. Malik, "Performance analysis of embedded software using implicit path enumeration", in *IEEE Trans. on CAD*, vol. 16, iss. 12, pp. 1477-1487, December, 1997.

[16] eCos real-time OS (`www.redhat.com/embedded/technologies/ecos`)

[17] Intel XScale microarchitecture (`http://developer.intel.com/design/xscale/`)

[18] T. A. Pering, T. D. Burd, and R. W. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms", in *Proc. ACM ISLPED*, pp. 76–81, August, 1998.

[19] D. Grunwald, P. Levis, and K. Farkas, "Policies for dynamic clock scheduling", in *Proc. OSDI*, 2000.

[20] Intel StrongARM processors (`http://developer.intel.com/design/strong/`)

[21] Transmeta Crusoe processor (`http://www.transmeta.com`)

[22] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A dynamic voltage scaled microprocessor system", in *IEEE Journal of Solid-State Circuits*, vol. 35, iss. 11, pp. 1571–1580, November, 2000.

[23] T. A. Pering, T. D. Burd, and R. W. Brodersen, "Voltage scheduling in the lpARM microprocessor system", in *Proc. ACM ISLPED*, pp. 96–101, 2000.

[24] A. Manzak and C. Chakrabarty, "Variable voltage task scheduling for minimizing energy or minimizing power", in *Proc. IEEE ICASSP*, June, 2000.

[25] C. M. Krishna and Y. H. Lee, "Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems", in *Proc. IEEE RTAS*, pp. 156–165, 2000.

[26] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low power embedded operating systems", in *Proc. 18th Symposium on Operating Systems Principles*, October, 2001.

[27] F. Gruian,"Hard real-time scheduling for low energy using stochastic data and DVS processor", in *Proc. ACM ISLPED*, August, 2001.

[28] J. Pouwelse, K. Langendoen, and H. Sips, "Energy priority scheduling for variable voltage processors", in *Proc. ACM ISLPED*, pp. 28–33, August, 2001.

[29] J. Luo and N. K. Jha, "Power conscious joint scheduling of periodic task graphs and aperiodic tasks in distributed real-time embedded systems", in *Proc. IEEE ICCAD*, pp. 357–364, November, 2000.

[30] J. Luo and N. K. Jha, "Battery aware static scheduling for distributed real-time embedded systems", in *Proc. ACM/IEEE DAC*, pp. 444–449, June, 2001.

[31] D. Zhu, R. Melhem, and B. Childers, "Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems", in *Proc. IEEE Real-Time Systems Symposium*, December, 2001.

[32] N. K. Jha, "Low power system scheduling and synthesis", in *Proc. IEEE ICCAD*, pp. 259–263, November, 2001.

[33] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment", in *Journal of ACM*, vol. 20, pp. 46-61, January, 1973.

[34] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-Completeness*. W. H. Freeman and Company, San Fransisco, CA, 1979.

[35] I. Hong, G. Qu, M. Potkonjak, and M. B.Srivastava, "Synthesis techniques for low-power hard real-time systems on variable voltage processors", in *Proc. IEEE RTSS*, pp. 178–187, 1998.

[36] A. Sinha and A. P. Chandrakasan, "Jouletrack: A web based tool for software energy pro ling", in *Proc. IEEE/ACM DAC*, June, 2001.

[37] J. L. W. V. Jensen, "Sur les fonctions convexes et les ingalits entre les valeurs moyennes", *Acta Math.*, vol. 30, pp. 175–193, 1906.

[38] W. Namgoong and T. H. Meng, "A high-ef cienc y variable-voltage CMOS dynamic dc-dc switching regulator", in *Proc. IEEE ISSCC*, pp.380-381, February, 1997.

[39] V. Gutnik and A. .P. Chandrakasan, "Embedded power supply for low-power DSP", in *IEEE Trans. on VLSI Systems*, vol.5, no.4, pp.425–435, December, 1997.

[40] A. Burns and A. Welling, *Real-Time Systems and their Programming Languages*. International Computer Science Series. Addison-Wesley, 1989.

[41] PARSEC parallel simulation language (`http://pcl.cs.ucla.edu/projects/parsec/`)

[42] A. Burns, K. Tindell, and A. Wellings, "Effective analysis for engineering real-time x ed priority schedulers", in *IEEE Trans. on Software Engineering*, vol.21, pp.475-480, May, 1995.

[43] N. Kim, M. Ryu, S. Hong, M. Saksena, C. Choi, and H. Shin, "Visual assessment of a real time system design: a case study on a CNC controller", in *Proc. IEEE RTSS*, December, 1996.

[44] C. Pereira, V. Raghunathan, S. Gupta, R. Gupta, and M. Srivastava, "A Software Architecture for Building Power Aware Real Time Operating Systems", Tech. Report #02-07, University of California, Irvine, March, 2002.

[45] Power Aware Distributed Systems project, UC Los Angeles and UC, Irvine. (`http://www.ics.uci.edu/ cpereira/pads`)

[46] Intel 80200 Evaluation Board (`http://developer.intel.com/design/xscale/`)

Energy Ef cient Wireless Packet Scheduling and Fair Queuing

Vijay Raghunathan, Saurabh Ganeriwal, Curt Schurgers, and **Mani Srivastava**
Networked and Embedded Systems Lab (NESL)
Department of Electrical Engineering
University of California, Los Angeles, CA 90095

Contact Author: VIJAY RAGHUNATHAN
Phone: (310) 206 4465
Fax: (310) 825 7928
E-mail: vijay@ee.ucla.edu

Other Authors:

SAURABH GANERIWAL	CURT SCHURGERS	MANI SRIVASTAVA
Phone: (310) 206 5698	Phone: (310) 206 4465	Phone: (310) 267 2098
Fax: (310) 825 7928	Fax: (310) 825 7928	Fax: (310) 794 1592
E-mail: saurabh@ee.ucla.edu	E-mail: curts@ee.ucla.edu	E-mail: mbs@ee.ucla.edu

Energy Efficient Wireless Packet Scheduling and Fair Queuing

Vijay Raghunathan, Saurabh Ganeriwal, Curt Schurgers, and **Mani Srivastava**
Networked and Embedded Systems Lab (NESL)
Department of Electrical Engineering
University of California, Los Angeles, CA 90095
{vijay, saurabh, curts, mbs}@ee.ucla.edu

Abstract

As embedded systems are being networked, often wirelessly, an increasingly larger share of their total energy budget is due to the communication. This necessitates the development of power management techniques that address communication subsystems, such as radios, as opposed to computation subsystems, such as embedded processors, to which most of the research effort thus far has been devoted. In this paper, we present techniques for energy efficient packet scheduling and fair queuing in wireless communication systems. Our techniques are based on an extensive slack management approach that dynamically adapts the output rate of the system in accordance with the input packet arrival rate. We use a recently proposed radio power management technique, Dynamic Modulation Scaling (DMS), as a control knob to enable energy-latency tradeoffs during wireless packet transmission. We first analyze a single input stream scenario, and describe a rate adaptation technique that results in significantly lower energy consumption (reductions of up to 10X), while still bounding the resulting packet delays. By appropriately setting the various parameters of our algorithm, the system can be made to traverse the energy-latency-fidelity tradeoff space. We extend our techniques to a multiple input stream scenario, and present E^2WFQ, an energy efficient version of the Weighted Fair Queuing (WFQ) algorithm for fair packet scheduling. Simulation results show that large energy savings can be obtained through the use of E^2WFQ, with only a small, bounded increase in worst case packet latency. Further, our results demonstrate that E^2WFQ does not adversely affect the throughput allocation (and hence, fairness) of WFQ.

I. INTRODUCTION

Conventional low power design techniques [1], [2], [3] and hardware architectures [4] only target digital computation systems, and relatively little work has been done for power optimization of the wireless communication subsystem. In many wireless embedded systems, communication energy dominates the energy consumed for computation [5], accentuating the need for radio power management methodologies. For example, in the wireless sensor nodes from Rockwell Inc. [6], transmitting one bit consumes 1500 to 2700 times [5] (depending on the transmission range) as much energy as executing one instruction. Therefore, in wireless devices, power management cannot just be limited to computation subsystems, such as processors, but has to extend to communication subsystems, such as radios, as well.

Radio power management is, however, complex, since the dependence of radio energy consumption on supply voltage and other circuit parameters is weak. Existing power management techniques that are effective for computation subsystems, such as Dynamic Voltage Scaling (DVS) [7], therefore do not result in the required energy savings. However, there exist similar control knobs on the radio that can be exploited for power management. The recently proposed technique of Dynamic Modulation Scaling (DMS) [8] uses the modulation level as an energy-speed control knob that can be fine-tuned to enable dynamic power-performance tradeoffs in the communication system (similar
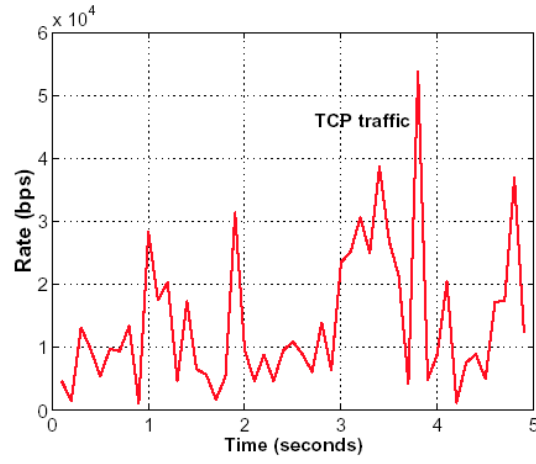
Figure 1. A 5 second workload trace at a network router

to what DVS provides for digital circuits [7]). DMS only reduces the radio's RF power, and does not target the power consumed in the radio electronics. Therefore, it is only applicable to medium and long range systems (with transmit distances of at least ten meters) such as wireless LANs, where the radio's RF power dominates the electronics power [8].

As is the case with variable voltage computation systems, significant energy benefits can be achieved in wireless communication systems by recognizing that peak performance (*i.e.*, service rate) is not always required. Since network traffic is characterized by a time varying workload requirement, energy can be saved by dynamically adapting the output transmission rate accordingly, through the use of DMS or other energy-latency tradeoff techniques. Figure 1 shows a 5 second snapshot of a real workload trace for a TCP traffic stream at a network router [9], exemplifying the inherent workload variability. For effective system-level power management, we need to develop algorithms that can exploit this variability by using control knobs such as DMS.

A. Paper contributions

In this paper, we present algorithms for energy efficient wireless packet scheduling and fair queuing of leaky bucket regulated input streams. Our algorithms are based on an extensive slack management approach, which exploits runtime slack created due to low link utilization, to dynamically adapt the output transmission rate of the system. Due to the use of DMS, transmitting at lower rates leads to energy savings. We first analyze a single input stream scenario, and describe a rate adaptation technique that reduces energy significantly, while bounding the resulting packet delays. Our scheme has various parameters, which can be adjusted to traverse the energy-latency-fidelity tradeoff space. We extend our techniques to a multiple input stream scenario, and present E^2WFQ[1], an energy efficient version of the Weighted Fair Queuing (WFQ) algorithm for fair packet scheduling. The use of E^2WFQ results in an energy aware packet scheduler that retains the fairness property of WFQ. Our previous work [12] showed the energy benefits of

[1]An initial version of our algorithm was presented in [11]

using DMS in the context of a deadline based real-time scheduling scheme. However, WFQ based packet schedulers are far more widely used than deadline based packet schedulers (both in wired and wireless environments) due to their desirable properties, which are elaborated further in Section II. Therefore, the techniques presented in this paper find applicability in enhancing the energy awareness of a much larger class of communication systems. To the best of our knowledge, this is the first work that attempts to incorporate energy awareness into rate based fair scheduling.

B. Related work

Several power management techniques have been proposed for the energy efficient design and operation of computation subsystems. One of the most effective techniques is DVS [7], where workload variability is exploited for energy savings by dynamically adjusting the processor's supply voltage and clock frequency to match the instantaneous performance requirement. Numerous energy aware task scheduling schemes have also been proposed, which utilize DVS to yield significant energy savings. Variable voltage task scheduling for workstation like environments has been explored in [13], [14], while the work described in [15], [16], [17], [18], [19], [20] deals with energy aware real-time task scheduling.

Unlike the large body of work that exists on variable voltage task scheduling, relatively little work has been done for energy efficient wireless packet scheduling. The problem of energy efficient transmission of a number of packets with deadlines has been addressed in [21], [22], using the concept of Dynamic Code Scaling. An API for power aware wireless communications is presented in [23] that enables applications to specify their requirements and constraints which the system takes into account while making power management decisions. Fair packet scheduling has been an active research topic in the networking community for a long time. Several fair scheduling schemes have been proposed for both wired (*e.g.,* Weighted Round Robin [24], Start-Time Fair Queuing [25], Worst-Case Fair Weighted Fair Queuing [26]) and wireless (*e.g.*, Channel-State Independent Wireless Fair Queueing [27], Wireless Packet Service [28], Server Based Fairness Approach [29]) environments. However, all of them are based on the concept of Generalized Processor Sharing [30], and its packetized version, Packet-by-Packet Generalized Processor Sharing [31] (PGPS, also known as WFQ [10]). Since network traffic usually displays a high temporal variation, the leaky bucket mechanism [30] is used to regulate and police input streams, and limit their variability. Finally, low power medium access protocols based on packet scheduling for wireless ATM networks were proposed in [32].

II. BACKGROUND

Since rate based packet scheduling schemes are significantly different from conventional CPU task scheduling schemes, we first review some of the basic concepts of rate based packet scheduling.

A. Generalized Processor Sharing (GPS)

A GPS scheduler divides the total link capacity C, among N input streams according to their service requirements. Each stream i is characterized by a weight ϕ_i, such that its service rate, g_i, is guaranteed to be [30][2]:

$$g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} \times C \tag{1}$$

GPS has the following attractive features: (i) Different input streams are well-isolated from each other (since each of them is allocated a guaranteed rate). Therefore, a malicious input stream cannot starve other streams of link bandwidth by generating large amounts of traffic. (ii) By varying the ϕ_i's, we have the flexibility of changing the fraction of the output link bandwidth that is allocated to each stream. As long as the combined average input rate of all the streams is less than C, any positive assignment of ϕ_i's yields a stable system.

B. Realizable implementations of GPS: WFQ

The GPS service model is an idealized one in which the traffic is considered to be infinitely divisible, and all streams are served simultaneously. Although GPS cannot be realized in practice, several real implementations of packet schedulers (for wired, as well as wireless environments) are based on it, and try to emulate the service provided by GPS as closely as possible.

Let F_p be the time at which a packet would complete service under GPS. Then, a good approximation of GPS is a scheme that serves packets with earliest F_p first. This scheme was proposed independently by two groups of researchers as Weighted Fair Queuing [10], and Packetized GPS [31], respectively. The difference in packet delays between GPS and WFQ was shown to be bounded by $\frac{L_{max}}{C}$, where L_{max} is the maximum packet size, and C is the output rate of the system [31].

C. Leaky bucket mechanism

The leaky bucket mechanism is often used to regulate and police the traffic in a network. This model is attractive since it restricts the incoming traffic in terms of average rate, as well as burstiness. Intuitively, the leaky bucket can be thought of as a container holding tokens, which is replenished with tokens at a rate of λ tokens per second. The container can hold at most B tokens. Whenever a packet of length L arrives, it is permitted to pass through only if the container holds at least L tokens. In that case, L tokens are pulled out of the container, and the packet is allowed to pass through. Thus, the container *leaks* tokens (if present) whenever packets arrive, hence the name *leaky bucket*. For a leaky bucket regulated stream, the amount of traffic, $A(\tau, t)$ that enters the network in time interval $(\tau, t]$ conforms to [30]:

$$A(\tau, t) \leq B + \lambda \times (t - \tau), \forall t \geq \tau \geq 0 \tag{2}$$

---

[2]Equation (1) assumes that all streams are backlogged (*i.e.*, have packets waiting to be sent). If a stream is not backlogged, it is not given any share of the link. Further, it does not contribute to the summation in Equation (1). Thus, GPS divides the output link capacity among only those streams that are backlogged.
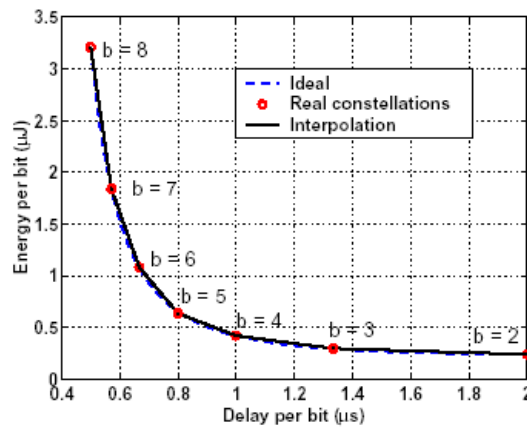
Figure 2. Energy-Delay tradeoff obtained through DMS

where λ characterizes the average rate of the stream, and **B** characterizes its burstiness. In real networks, input traffic streams pay for the network resources according to the quality of service they desire, and the traffic description that they provide up front. The traffic sources ensure that the traffic they generate conforms to the specifications. In addition, the network provider uses a leaky bucket to police the incoming traffic, and enforce the specifications if traffic sources misbehave and deviate from agreed behavior. When the input sources are constrained by leaky buckets, it is possible to give a worst case queuing delay guarantee using GPS (and therefore, WFQ).

Theorem 1: *If the input traffic of stream is constrained using a leaky bucket with parameter (B_i, λ_i), and g_i is its guaranteed rate, then the maximum delay for a packet of stream , under GPS scheduling is given by [31]:*

$$D_i \leq \frac{B_i}{g_i}$$

(3)

D. Dynamic Modulation Scaling

We next review the basics of DMS, which was introduced in [8]. To transmit information, bits are coded into channel symbols. The number of bits per symbol is given by the modulation level **b** . This **b** is the radio control knob that allows DMS to trade off energy versus delay. The average time to transmit one bit is given by Equation (4), where R_S is the symbol rate in number of symbols sent over the channel per second.

$$T_{bit} = \frac{1}{b \times R_S}$$

(4)

Although DMS is applicable to other scalable modulation schemes as well, we focus on Quadrature Amplitude Modulation (QAM) as it is both efficient and easy to implement [33]. The energy consumed for transmitting one bit is given by [8]:

$$E_{bit} = C_S \times \frac{2^b - 1}{b} + C_E \times \frac{1}{b}$$

(5)

211

The rst term gives the energy consumed in the radio front-end for generating the electro-magnetic waves that carry the information. Parameter C_S depends on the radio implementation, the wireless channel, the transmit distance, and the required error performance. Strictly speaking, it is also a function of b, but a very weak one [8], and therefore, we assume it to be a constant. The rest of the radio energy consumption is lumped into the second term of Equation (5), where C_E depends on the radio implementation. Although Equation (5) is only exact for *even* integer values of b, it is also a reasonable approximation when b is *odd*. In addition, a packet can be split into two parts, each with a different modulation. In this case, the average energy and delay per bit for the packet as a whole are a linear interpolation between the corresponding values of the two modulation levels.

Figure 2 plots the energy versus delay for the following parameter values: R_S = 250 KHz, C_S = 100 nJ, and C_E = 180 nJ. The values of C_S and C_E are extracted from [34], which describes the implementation of an adaptive QAM system[3]. Figure 2 shows the operating points that correspond to real constellations and those that are obtained through interpolation. The curve labeled *Ideal* is calculated from the above equations. Each modulation scheme has a b_{min}, which is equal to 2 for QAM. The maximum modulation b_{max} is only bounded by implementation constraints. In our work, we choose b_{max} equal to 8, and scale the modulation level with a granularity of 0.5 bits/symbol. When the sender changes the modulation, the receiver needs to be told. To avoid a complicated sender-receiver protocol for modulation changes, and to limit the associated overhead, we restrict these changes to be done only at the start of packet transmissions. This nite time-granularity is a crucial difference between DMS and dynamic voltage scaling [7].

III. ENERGY EFFICIENT SCHEDULING ALGORITHM

Having explained the basics of rate based fair scheduling and DMS, we next present our algorithm that uses DMS to incorporate energy awareness into wireless packet scheduling. We rst discuss the case where there is a single input stream, and then extend it to a fair queuing scenario where there are multiple input streams that are time-multiplexed onto a single outgoing transmission link.

A. System model

Figure 3 shows a generic block diagram of our system, which consists of a leaky bucket regulated input stream being serviced by an energy ef cient scheduler. The scheduler is connected to a radio that supports DMS. As shown later in Section III.H, this model can be extended to multiple input streams sharing a single outgoing communication link. The leaky bucket parameters are given by the tuple (λ, B), which denote the average token arrival rate, and the bucket size, respectively. Input traf c arrives at a rate $A(t)$, and the adaptive modulation radio ensures that the scheduler operates at a time varying output rate of $R(t)$.

---

[3]Since the system in [34] was designed for high speed rather than low power applications, it is likely that these numbers can be reduced further through the use of dedicated circuit design techniques.
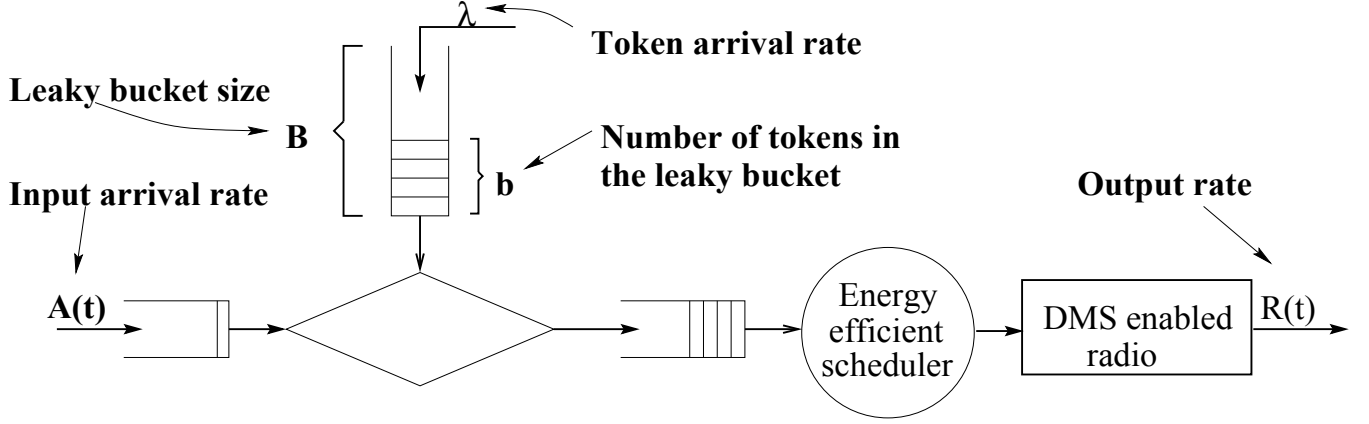
Figure 3. System model showing a leaky bucket regulated stream being serviced

B. Energy saving opportunities

In many practical scenarios, the average input rate of a stream is lower than the peak output link capacity, R_{max}, resulting in a low link utilization. In addition, packet lengths may often be smaller than the maximum value, thereby reducing the utilization even further. As a result, it is possible to operate at an instantaneous output rate, $R(t)$ that is lower than the maximum rate for most of the time, without adversely affecting the link performance. So, instead of operating at R_{max}, and shutting down the radio when idle, we slow transmissions down to a rate $R(t)$ through the use of DMS. As can be seen in Figure 2, the energy-delay curve is convex, which implies that slowing down the radio is more energy efficient than shutdown (similar to what is observed in DVS [7]). Note that this also reduces the overhead associated with shutting down and restarting the radio, which can often be significant [35].

C. Overview of the problem

The goal of our energy efficient packet scheduling algorithm is to adapt the output rate $R(t)$ to match the instantaneous workload. At the same time, we would like to bound the performance impact that may result. Due to the convexity of the energy-speed curve and Jensen's inequality $\left(E(\overline{r}) \leq \overline{E(r)} \right)$ [36], workload averaging results in higher energy savings. In fact, maximum energy savings would be obtained if we operated at the long term average input rate, thereby smoothing out all the input workload variations. However, buffering the input variations leads to a performance penalty due to an increase in packet delays. Therefore, the crux of the problem reduces to determining the degree of buffering (*i.e.*, how much workload averaging to perform) while bounding the increase in packet delays.

D. Computing the instantaneous output rate

We use an elaborate slack management technique to accurately compute the minimum possible transmission rate for each packet such that the performance impact is within acceptable limits. In our technique, each packet is associated with some slack information, which is indicative of how much the packet can be slowed down. The pseudo-code for computing this slack information is given in Figure 4. Whenever a packet arrives, all the packets ahead of it in the queue are inspected to see how much slack can be allocated to the newly arrived packet while still meeting its deadline.

213

```
Procedure COMPUTE_SLACK
/* Executes whenever a new packet arrives.
 * The packet is at the k^th position in the queue.  L_i denotes the
 * length of the i^th packet, and s_i denotes the slack associated with
 * it.  T_remaining is the remaining transmission time of the packet
 * currently being transmitted, T_stop is its time of completion,
 * and T is the current time.  D and R_max denote the deadline,
 * and maximum output rate, respectively.  x is the time required
 * to transmit all the packets in the queue and all the available
 * slack.  */
{
  if (radio is currently idle)
    T_remaining = 0;
  else
    T_remaining = T_stop - T;
  s_k = 0;
  x = [Σ_{i=0}^{k} (L_i + s_i)] × 1/R_max + T_remaining;
  if (x ≤ D)                       /* We can tolerate more slack for */
    s_k = (D - x) × R_max;         /* the newly arrived packet */
  else{
    j = k;                         /* We need to delete slack */
    y = x - D;                     /* from preceding packets */
    while (y > 0) {
      if (s_j ≥ y) {
        s_j = s_j - y;
        y = 0;
      } else {
        y = y - s_j;
        s_j = 0;
        j = j - 1;
      }
    }
  }
}
```

Figure 4. Pseudo-code for computing the slack associated with a newly arrived packet

In certain cases, slack might have to be deleted from preceding packets in the queue to ensure that the newly arrived packet meets its deadline. The slack deletion proceeds in reverse order of packet arrival to retain maximum flexibility in rate-adaptation. The packet is then tagged with this available slack information. Thus, every packet, i, in the queue has a certain slack, s_i, associated with it.

The slack information is used in the actual rate computation algorithm itself. The output rate computation for a packet is done when it reaches the head of the queue. We can safely utilize all the slack associated with the Head Of Line (HOL) packet while guaranteeing the in-time transmission of all the packets currently in the queue. However, this slack does not account for future packet arrivals. If we use all the available slack for transmitting the HOL packet, and a burst of packets arrive immediately after transmission has begun, it is quite possible that some of the packets in the burst will miss their deadlines even if we switch to maximum speed immediately after the current transmission.

214

Therefore, we need to set an additional contraint on the amount of slack that can be utilized, to guarantee deadline satisfaction for the worst case pattern of future packet arrivals. The worst case arrival pattern is a burst that drains the leaky bucket completely. Under this arrival pattern, the amount of slack (s) that can be utilized for the HOL packet, while guaranteeing no deadline misses in the future, is bounded by:

$$s \leq D \times R_{max} - b - \sum_{i=0}^{k} L_i \qquad (6)$$

where b is the number of tokens currently in the leaky bucket, k is the number of packets in the queue, and L_i is the length of the ith packet in the queue. D and R_{max} denote the deadline and the maximum output rate, respectively. Obviously, the other constraint on the amount of usable slack is ($s \leq s_0$), where s_0 is the total available slack for the HOL packet. Any slack distribution approach that satis es these two constraints can be used to allocate slack to the HOL packet. We have used a slack distribution approach that tries to perform as much workload averaging as possible, since averaging reduces energy consumption, as mentioned earlier. After a certain amount of slack has been used to compute the output rate, R, to transmit the HOL packet, the queue status has to be updated. The pseudo-code of our rate computation and slack updation algorithms is given in Figure 5.

E. Setting the output link speed

Although the maximum output link rate is R_{max}, the instantaneous required rate by the input stream is R. This means that if ($R < R_{max}$), packet transmissions can be slowed down to just meet the instantaneous requirement, thus saving energy. The new modulation level for the outgoing packet is given by:

$$b_{instantaneous} = \frac{R}{R_{max}} \times b_{max} \qquad (7)$$

F. Packet delay bound

To quantify the impact of our scheme on packet latencies, we introduce a new parameter Δ, which indicates the maximum latency hit that the input stream is willing to tolerate in exchange for energy ef cienc y. As we will see shortly, Δ determines the system's response to workload variations, by deciding the degree of buffering.

De nition 1: *We de ne Δ to be the additional latency that packets of the input stream are willing to tolerate, compared to a scheduling scheme where the radio always operates at the maximum rate.*

A x ed rate scheduling scheme would always transmit at the maximum rate R_{max}. Using properties of the leaky bucket mechanism, it can be shown [31] that such a scheme provides a packet delay bound of $\frac{B}{R_{max}}$. Therefore, from the de nition of Δ, the maximum delay of a packet under the energy ef cient scheduling scheme described in Figure 4 and Figure 5, is given by:

$$D \leq \left(\frac{B}{R_{max}} + \Delta \right) \qquad (8)$$

where B is the leaky bucket size, and R_{max} is the maximum output link capacity (*i.e.*, at a modulation level b_{max}).

Procedure COMPUTE_TRANSMISSION_RATE

```
/* Executes whenever the HOL packet begins transmission.
 * There are k packets in the queue.  L_i denotes the
 * length of the i^th packet, and s_i denotes the
 * slack associated with it.  R denotes the transmission
 * rate of the HOL packet.  D and R_max denote the
 * deadline and maximum output rate, respectively.  */
```

$\{$

$R = 0;$

for $(j = 0; j \leq k; j + +)\{$

$\quad R_j = R_{max} \times \dfrac{\sum_{i=0}^{i=j} L_i}{\sum_{i=0}^{i=j} (L_i + s_i)};$

$\quad$**if** $(R_j > R)$

$\quad\quad R = R_j;$

$\}$

$s = L_0 \times \left(\frac{R_{max}}{R} - 1\right);$

if $\left(s > \left(D \times R_{max} - b - \sum_{i=0}^{k} L_i\right)\right)$

$\quad s = \left(D \times R_{max} - b - \sum_{i=0}^{k} L_i\right);$

$R = R_{max} \times \frac{L_0}{L_0 + s};$

$\}$

Procedure UPDATE_SLACK_INFORMATION

```
/* Executes whenever the HOL packet begins transmission.
 * There are k packets in the queue.  R is the transmission
 * rate of the HOL packet.  T is the current time, and T_stop
 * is when the HOL packet will finish transmission.  L_i and
 * s_i are the length, and slack associated with the i^th packet,
 * respectively.  */
```

$\{$

$T_{stop} = T + \frac{L_0}{R};$

$L_0 = L_1;$

$s_0 = s_0 - s + s_1;$

for $(i = 0; i <= k; i + +)\{$

$\quad L_{i-1} = L_i;$

$\quad s_{i-1} = s_i;$

$\}$

$k = k - 1;$

$\}$

Figure 5. Pseudo-code for computing the transmission rate of the HOL packet, and updating slack information

G. Analyzing the system's response

We next analyze the system's response to a change in workload to highlight the effect of Δ. Let the input rate of the stream be equal to R_{old}. If we set the output rate to be equal to R_{old}, then the queue soon reaches a steady state, and the number of packets in the queue remains constant. Let the system be at such a steady state at time t, and the number of packets in the queue be $y(t)$. In this state, any packet that arrives, sees exactly $y(t) - 1$ packets ahead of it in the queue, each of length, say L. All these packets (including the one that just arrived) have to complete transmission within a time Δ (since the worst case arrival pattern of future packets is potentially a burst of size B). Therefore, the
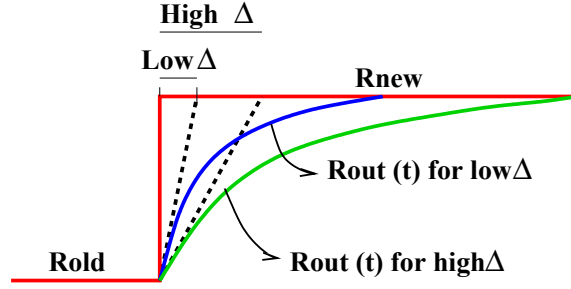
216

Figure 6. The effect of Δ on the output rate

output rate, $R_{out}(t)$ is given by:

$$R_{out}(t) = \frac{y(t) \times L}{\Delta} = R_{old} \tag{9}$$

Now, let the input rate increase abruptly to R_{new}. We want to analyze how fast the system stabilizes to its new steady state. After a time δt, the number of packets in the queue becomes:

$$y(t + \delta t) = y(t) + \frac{R_{new} - R_{out}(t)}{L} \times \delta t \tag{10}$$

The new output rate will be given by $R_{out}(t + \delta t)$, which after some simple substitutions, can be written as:

$$R_{out}(t + \delta t) = R_{out}(t) + \frac{R_{new} - R_{out}(t)}{\Delta} \times \delta t \tag{11}$$

Therefore, the rate at which the output rate changes is given by:

$$\frac{\delta R_{out}}{\delta t} = \frac{1}{\Delta} \times (R_{new} - R_{out}(t)) \tag{12}$$

Solving this differential equation, and applying the initial condition $R_{out}(t) = R_{old}$, we get:

$$R_{out}(t) = R_{new} + (R_{old} - R_{new}) \times e^{\frac{-t}{\Delta}} \tag{13}$$

The evolution of $R_{out}(t)$ is shown in Figure 6. Therefore, it can be concluded that Δ is the time constant of the rst-order input response of the system. A high value of Δ means that the system takes longer to switch to the new steady state, which implies that steep workload transients are ltered out. While this increases the energy savings, the maximum impact on packet delay also increases (see Equation (8)). On the other hand, a low value of Δ means that the system is sensitive to changes in the input rate, and performs less workload averaging. This reduces the energy savings, but also results in a smaller impact on packet delays. The best choice of Δ thus depends on the allowable delay, and the input rate variability. Another way of explaining the effect of Δ is to consider the scheduler as a low pass lter performing some workload ltering on the input traf c. Then, Δ is inversely proportional to the bandwidth of the low pass lter . If Δ is high, the low pass lter has a very low passband, and hence permits only very low frequency workload transients to pass through.

H. Handling multiple input streams: E^2WFQ

Next, we consider a WFQ based scheduling model, and show how the algorithm described in the previous subsections can be extended to the case where multiple input streams are multiplexed onto a single output transmission link of maximum capacity C. Our enhanced scheduling technique, which we call E^2WFQ, can be used either for multiple streams sharing a point-to-point link, or for multiple streams destined to different one-hop neighbors of a wireless node (*e.g.*, a base station sending data to multiple clients). Each input stream i is regulated by a leaky bucket with parameters (B_i, λ_i). Therefore, stream i produces packets at an average rate of λ_i. The average rate at which packets arrive at the scheduler is $\lambda = \sum_{i=1}^{N} \lambda_i$, where N is the total number of streams sharing the output link. Stream i is allocated a weight ϕ_i, which leads to a corresponding guaranteed rate, g_i.

Taking advantage of the ow isolation property of WFQ, we can treat each input stream independently, and compute its instantaneous rate requirement. The algorithm presented in Figures 4 and 5 still holds, with R_{max} being replaced by g_i, the guaranteed rate of stream i. At time t, if $R_{out,i}(t)$ is the output rate for stream i, as computed by our algorithm, the instantaneous required rate (R) of the entire system is obtained by summing the output rate over all the streams. It is given by:

$$R = \sum_{i=1}^{N} R_{out,i}(t) \tag{14}$$

If C is the maximum output link capacity, the modulation level for outgoing packets is then computed using the following equation:

$$b_{instantaneous} = \frac{R}{C} \times b_{max} \tag{15}$$

I. Delay guarantee provided by E^2WFQ

Through the choice of the parameter Δ, our scheduling scheme provides the following delay bound for packets.

Theorem 2: *The maximum delay of a packet of stream i, under the E^2WFQ scheduling scheme is given by:*

$$D_i^* \leq \left(D_i + \Delta + \frac{L_{max}}{C_{min}} \right) \leq \left(\frac{B_i}{g_i} + \Delta + \frac{L_{max}}{C_{min}} \right) \tag{16}$$

where C_{min} is the output link capacity at a modulation level b_{min}.

As before, increasing the parameter Δ increases the maximum packet latency incurred by the input streams, while decreasing the energy consumption due to increased workload averaging.

IV. SIMULATION RESULTS

We have carried out a number of simulations to evaluate our techniques. In the following subsections, we describe our simulation framework, and present our results.
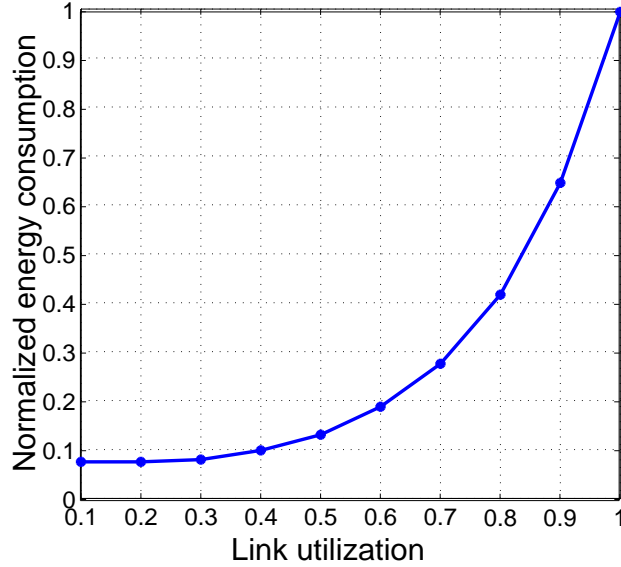
218

Figure 7. Normalized energy consumption as a function of link utilization

A. Simulation framework

To evaluate the performance of our algorithm, we built a discrete event simulator using PARSEC [37], a C based parallel simulation language. We considered a leaky bucket regulated input stream being serviced by an energy ef cient scheduler connected to an output link of capacity R_{max} = 500 Kbit/s The maximum packet size was set to be L_{max} = 1000 bits, and the stream was leaky bucket constrained with a maximum burstiness parameter $B = 10^5$ bits (corresponding to a maximum burst size of 100 packets). For our values, D was equal to 0.2 seconds. The average rate of the input was varied to change the link utilization. As explained in the previous section, the choice of Δ offers a tradeoff between energy savings and queuing delay. Unless explicitly speci ed, all the experiments used a value of $\Delta = 0.05$ seconds. Finally, when multiple input streams are present, our scheme does not require Δ to be a global parameter. Each stream can choose a different value of Δ, depending on the packet delays acceptable to it.

B. Discussion of the results

Figure 7 shows the energy consumed by our algorithm, normalized against the energy consumed by an energy unaware scheme, for varying values of output link utilization. This curve was generated with no burstiness present in the input traf c and packet lengths all set to 1000 bits, although we show later that any burstiness in the input traf c does not affect our scheme. As expected, the energy consumption decreases as the link utilization drops. Our algorithm reduces to a x ed speed algorithm at a utilization of one, since there are no opportunities for slowing down. Figure 8 shows the effect of input traf c burstiness on the performance of our algorithm. As can be seen in Figure 8(a), the energy savings obtained through our scheme are independent of the burstiness of the input traf c. By explicitly incorporating information about the state of the leaky bucket during rate adaptation, our scheme is able to offer signi cant energy savings even for highly bursty traf c. Figure 8(b) plots the resulting packet delays as a
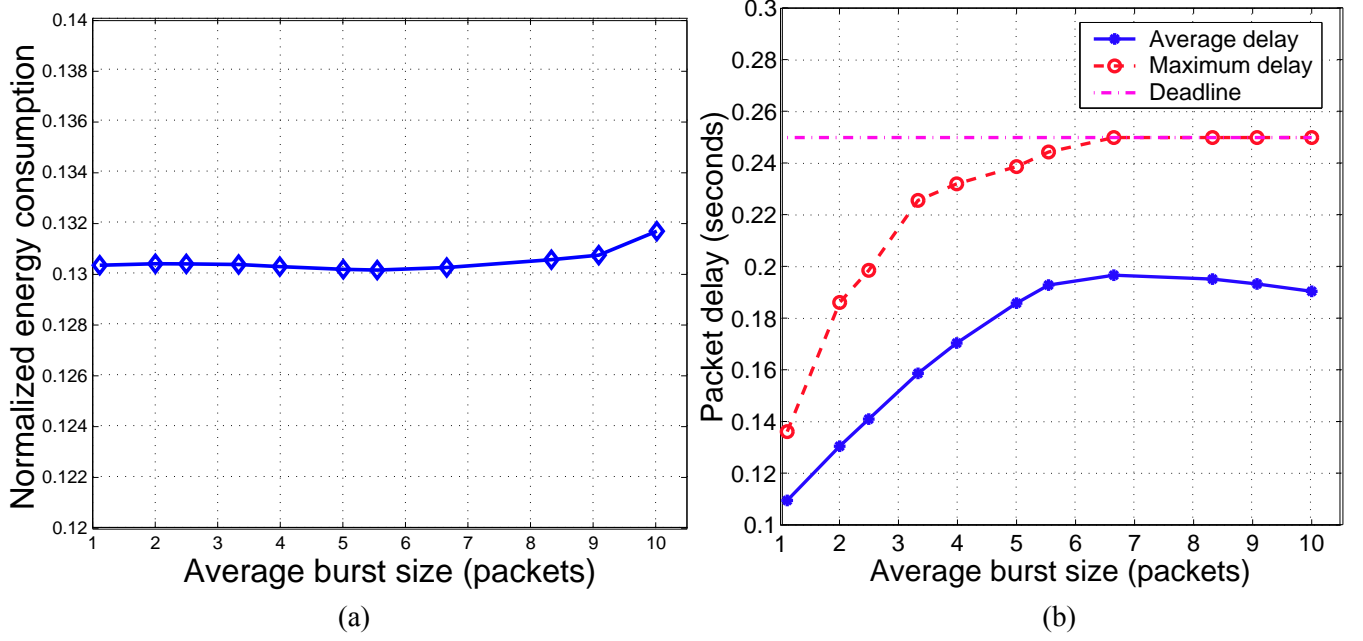
Figure 8. (a) Normalized energy consumption as a function of input burstiness, and (b) Packet delays as a function of burstiness
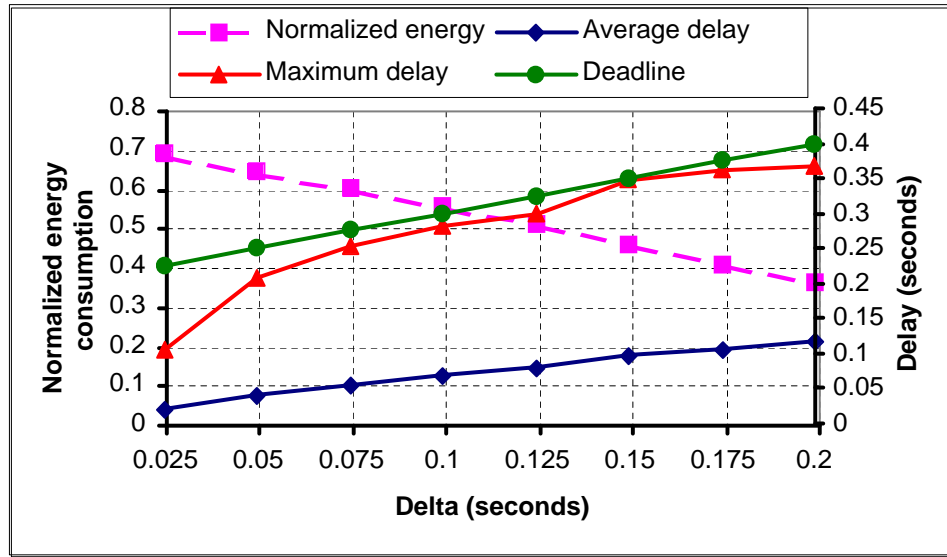

Figure 9. Normalized energy consumption, and average and maximum packet delays , as a function of Δ

function of the input traffic burstiness. As expected, when the burstiness increases, both the maximum and average packet delays also increase. However, note that all packets complete transmission before their deadlines, thereby avoiding any timing violations in the system. Figures 7 and 8 show that the use of our scheduling scheme decreases the energy consumption of the system by up to a factor of 10X, with a small, bounded increase in packet latency.

As was mentioned in Section III, the parameter Δ determines the amount of buffering that our scheme performs. Therefore, Δ can be used as a control knob to adjust the operating point on an energy-latency tradeoff curve. We performed an additional experiment to demonstrate the effect of the parameter Δ, where we observed the energy consumption and packet delays, as Δ was varied. The results are shown in Figure 9. Increasing Δ increases the
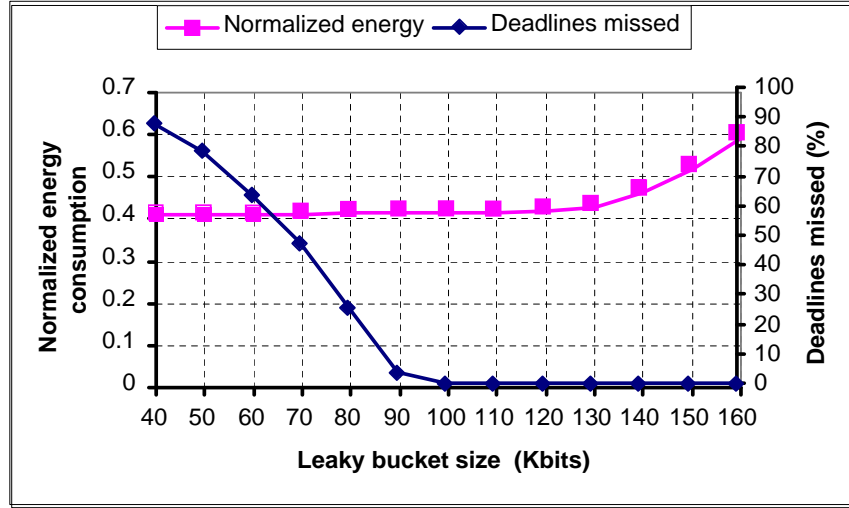
Figure 10. Impact of the leaky bucket size on energy consumption and system fidelity
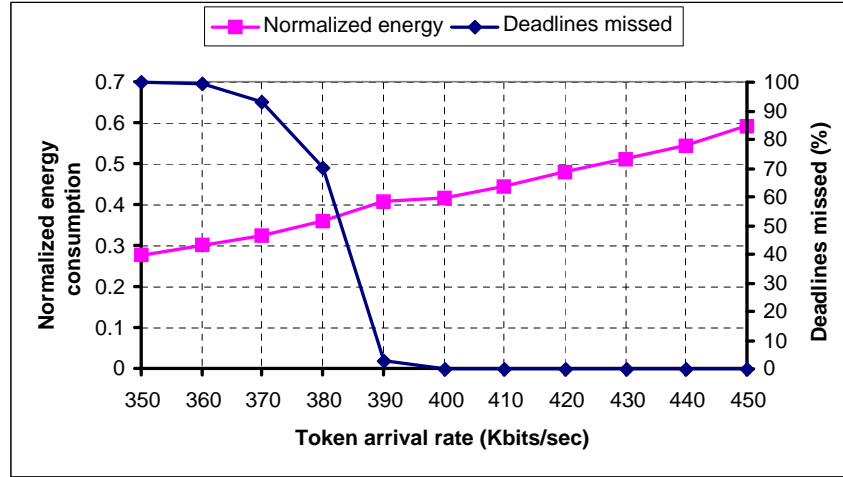

Figure 11. Impact of the token arrival rate on energy consumption and system fidelity

allowable latency (*i.e.,* deadline) of a packet. As can be seen from the figure, the maximum and average delays also increase, trading off the extra latency for additional energy savings. Therefore, the energy consumption decreases with increasing Δ. Thus, by varying Δ, the system can operate at different points in this energy-performance plane.

Our next set of experiments demonstrate the effect of the leaky bucket parameters on the system fidelity. Here, the link utilization and maximum input burstiness were fixed at a value of 0.8, and 100 packets, respectively, and the leaky bucket parameters were varied. The results are shown in Figure 10 and Figure 11. If the average rate and burstiness of the input traffic are more than the corresponding leaky bucket parameters (*i.e.*, the leaky bucket parameters are under-specified), it results in a degradation in the quality of service received by some packets of the flow, thereby leading to deadline misses. Figure 10 plots the energy consumption and the percentage of packets missing their deadlines as a function of the leaky bucket size. It is evident from the figure that if the leaky bucket size is under-specified (*i.e.,* less than 100 Kbits corresponding to the maximum burst size of 100 packets), deadline misses begin to occur. Figure 11 shows the effect of the token arrival rate at the leaky bucket on the energy consumption and percentage of deadline
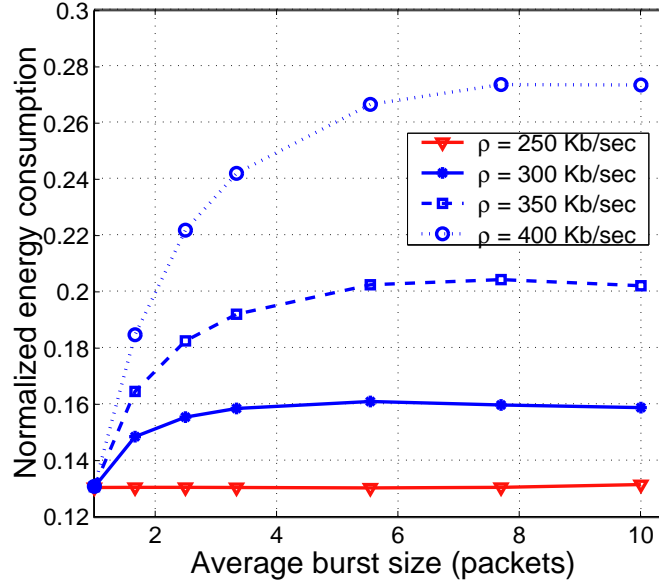
Figure 12. Effect of input burstiness on the energy consumption for different values of token arrival rate

misses. Again, at token arrival rates that are lower than the average input rate of 400 Kbits/sec (corresponding to a link utilization of 0.8), the leaky bucket chokes the input stream, resulting in a large number of deadline misses. Finally, Figure 12 shows the impact of input traffic burstiness on the energy consumption of our scheme, for different values of the leaky bucket token arrival rate. The average input rate was set to 250 Kbits/sec, resulting in a link utilization of 0.5. As can be seen in the gure, when the token rate is correctly speci ed, our scheme is not impacted by burstiness in the input traf c. However, if the token rate is under-speci ed, the energy consumption of the system increases with increasing burstiness. These curves highlight the importance of correctly setting the leaky bucket parameters, if high system delity is required. In real networks, input traf c streams pay for the network resources according to the quality of service they desire, and the traf c description that they provide up front to the network service provider. The network provider polices the incoming traf c using a leaky bucket with parameters set according to the traf c description, and enforces the speci cations if traf c sources misbehave and deviate from agreed behavior. Therefore, either the source of the input traf c or the network service provider should accurately estimate the statistics of the input traf c, which can then be used to set the leaky bucket parameters accordingly. Further, for increased energy scalability, the leaky bucket parameters can be deliberately tweaked to result in a gradual decrease in system delity as the system's energy supply runs out.

Our nal set of experiments involved comparing the performance of our scheme, E^2WFQ with unmodi ed WFQ. For this purpose, we considered a scenario where two input streams (henceforth referred to as *Flow 1* and *Flow 2*) shared an output link of capacity 500 Kbps. The o ws were allocated weights in the ratio 2:3, leading to guaranteed rates of 200 Kbps and 300 Kbps, respectively. Figures 13(a) and 13(b) plot the throughput received by the two o ws as well as the aggregate throughput, as a function of time, using WFQ and E^2WFQ, respectively. Initially, the o ws do not require their maximum guaranteed bandwidth, and hence are allocated only so much as they currently require. As
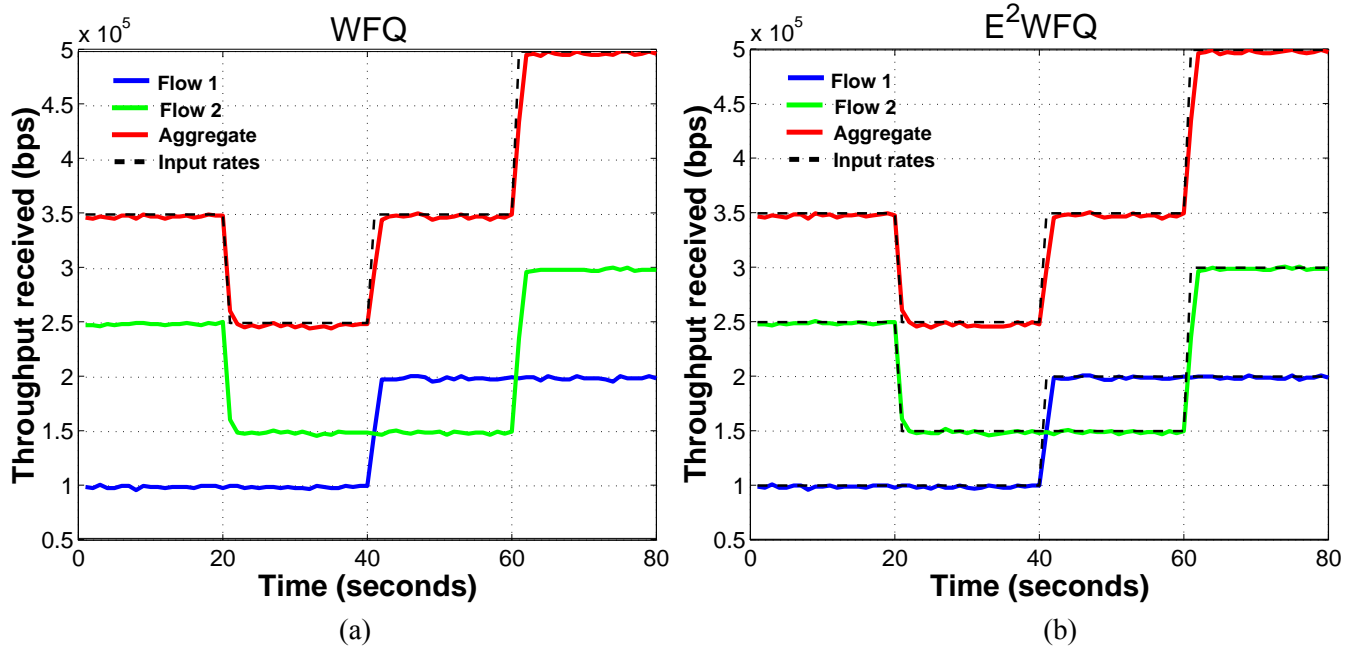
WFQ

E²WFQ

Figure 13. Throughput received by *Flow 1* and *Flow 2* using (a) WFQ, and (b) E^2WFQ. The two figures are identical, indicating that E^2WFQ does not affect the throughput allocation (and hence, fairness) of WFQ.

can be seen from the figure, the throughput allocation obtained by using E^2WFQ is identical to that obtained using WFQ. This shows that our scheme retains the fairness property of WFQ. Figure 14 plots the energy consumption of the E^2WFQ system over 1 second intervals, normalized to the case when WFQ is used. As can be seen, our scheme results in significant reductions in energy consumption compared to the WFQ case. The curve labelled *Ideal* denotes the normalized energy consumption that would have resulted if the system had just one input flow with utilization equal to the aggregate utilization of *Flow 1* and *Flow 2*. Note that E^2WFQ performs almost as well as the single flow case, which indicates that our scheme is able to distribute the energy benefits obtained due to the low link utilization of one flow, among the other flows while still maintaining flow isolation for throughput purposes.

V. CONCLUSIONS

In wireless embedded systems, communication energy is increasingly becoming the dominant component of total energy consumption. Dynamic power management techniques therefore, should also address communication subsystems such as radios, as opposed to only computation subsystems such as embedded processors. DMS is a technique that offers a power-speed control knob for the radio, and can thus be used for communication power management. However, higher level techniques are needed that can exploit this control knob to enable system level energy-latency tradeoffs. In this paper, we have targeted the packet scheduling process and investigated avenues for making it energy aware. We have presented an energy efficient packet scheduling algorithm for leaky bucket regulated traffic streams, which provides significant energy savings (up to a factor of 10X) with only a small, bounded increase in worst case packet delay. We have extended our scheme to a multiple input stream scenario, and have enhanced the Weighted Fair Queuing algorithm for energy efficiency. Our techniques find applicability for medium and long range wireless
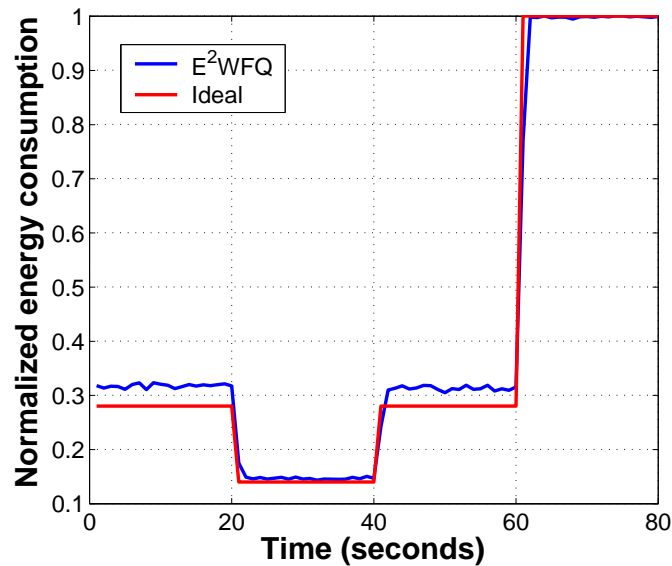
Figure 14. Normalized energy consumption of E^2WFQ

communication systems (such as wireless LANs), where DMS is effective in reducing energy consumption, due to the radio's RF power dominating the power consumed in the radio electronics. We believe that our work is a rst step towards the percolation of the more mature computation power management techniques into the communications realm.

References

[1] J. Rabaey and M. Pedram, *Low Power Design Methodologies*. Kluwer Academic Publishers, Norwell, MA, 1996.

[2] A. Raghunathan, N. K. Jha, and S. Dey, *High-level Power Analysis and Optimization*. Kluwer Academic Publishers, Norwell, MA, 1998.

[3] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, Norwell, MA, 1997.

[4] A. P. Chandrakasan and R. W. Brodersen, *Low Power CMOS Digital Design*. Kluwer Academic Publishers, Norwell, MA, 1996.

[5] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy aware wireless microsensor networks", *IEEE Signal Processing Magazine*, vol. 19, iss. 2, pp. 40–50, March 2002.

[6] Rockwell Inc. (`http://wins.rsc.rockwell.com`)

[7] T. A. Pering, T. D. Burd, and R. W. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms", *Proc. ACM ISLPED*, pp. 76–81, 1998.

[8] C. Schurgers, O. Aberthorne, and M. B. Srivastava, "Modulation scaling for energy aware communication systems", *Proc. ACM ISLPED*, pp. 96–99, 2001.

[9] The Internet Traf c Archive (`http://ita.ee.lbl.gov`).

[10] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorthm", *Internet Res. and Exper.*, vol. 1, 1990.

[11] V. Raghunathan, S. Ganeriwal, C. Schurgers, and M. B. Srivastava, "E^2WFQ: An energy ef cient fair scheduling policy for wireless systems", *Proc. ACM ISLPED*, pp. 30–35, 2002.

[12] C. Schurgers, V. Raghunathan, and M. B. Srivastava, "Modulation scaling for real-time energy-aware packet scheduling", *Proc. IEEE GLOBECOM*, pp. 3653–3657, 2001.

[13] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy", *Proc. First Symp. on OSDI*, pp. 13–23, 1994.

[14] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU", *Proc. ACM MOBICOM*, pp. 13–25, 1995.

[15] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors, *Proc. ACM ISLPED*, pp. 197–202, 1998.

[16] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processors", *Proc. IEEE ICCAD*, pp. 653–656, 1998.

[17] Y. Shin and K. Choi, "Power conscious x ed priority scheduling for hard real-time systems", *Proc. ACM DAC*, pp. 134–139, 1999.

[18] V. Raghunathan, P. Spanos, and M. B. Srivastava, "Adaptive power- delity in energy-aware wireless systems", *Proc. IEEE RTSS*, pp. 106–115, 2001.

[19] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low power embedded operating systems", *Proc. SOSP*, pp. 89–102, 2001.

[20] F. Gruian, "Hard real-time scheduling for low energy using stochastic data and DVS processors", *Proc. ACM ISLPED*, pp. 46–51, 2001.

[21] B. Prabhakar, E. U. Biyikoglu, and A. El Gamal, "Energy-ef cient transmission over a wireless link via lazy packet schedul-ing", *Proc. IEEE INFOCOM*, pp. 386–394, 2001.

[22] A. El Gamal, C. Nair, B. Prabhakar, E. U. Biyikoglu, and S. Zahedi, "Energy-ef cient scheduling of packet transmissions over wireless networks", *Proc. IEEE INFOCOM*, pp. 1773–1782, 2002.

[23] R. Min and A. P. Chandrakasan, "A framework for energy scalable communication in high-density wireless networks", *Proc. ACM ISLPED*, pp. 36–41, 2002.

[24] M. Shreedhar and G. Varghese, "Ef cient fair queueing using de cit round robin", *Proc. IEEE SIGCOMM*, pp. 231–242, 1995.

[25] P Goyal, H. M. Vin, and H. Cheng, "Start-time fair queuing: A scheduling algorithm for integrated servicespacket switching networks", *Proc. IEEE SIGCOMM*, pp. 157–168, 1996.

[26] J. C. R. Bennett and H. Zhang, "WF^2Q : Worst-case fair weighted fair queueing, *Proc. IEEE INFOCOM*, pp. 120–128, 1996.

[27] P. Lin, B. Bensaou, Q. L. Ding, and K. C. Chua, "A wireless fair scheduling algorithm for error-prone wireless channels", *Proc. ACM WoWMOM*, pp. 11–20, 2000.

[28] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks", *Proc. IEEE SIGCOMM*, pp. 63–74, 1997.

[29] P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks", *Proc. ACM MOBICOM*, pp. 1–9, 1998.

[30] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, 1997.

[31] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to o w control in integrated services networks: The single-node case", *IEEE Trans. on Networking*, vol. 1, no. 3, pp. 344-357, June 1993.

[32] K. M. Sivalingam, M. Srivastava, P. Agrawal, and J. C. Chen, "Low power access protocols based on scheduling for wireless and mobile ATM networks", *Proc. IEEE Universal Personal Communications Conference*, pp. 420–433, 1997.

[33] J. G. Proakis, *Digital Communications*. McGraw Hill, 1989.

[34] K. Cho and H. Samueli, "A 8.75-MBaud Single-Chip Digital QAM Modulator with Frequency-Agility and Beamforming Diversity", *Proc. IEEE Custom Integrated Circuits Conference*, pp. 27–30, 2000.

[35] A. Y. Wang, S. H. Cho, C. G. Sodini, and A. P. Chandrakasan, "Energy ef cient modulation and MAC for asymmetric RF microsensor systems", *Proc. ACM ISLPED*, pp. 106–111, 2001.

[36] J. L. W. V. Jensen, "Sur les fonctions convexes et les ingalits entre les valeurs moyennes", *Acta Math.*, vol. 30, pp. 175–193, 1906.

[37] PARSEC parallel simulation language (`http://pcl.cs.ucla.edu/projects/parsec`)